Universidad Carlos III de Madrid
www.uc3m.es

**TRABAJO FIN DE GRADO**

# ESTUDIO EXPERIMENTAL Y NUMÉRICO DEL FLUJO OSCILATORIO ALREDEDOR DE DOS CILINDROS

**Autor: Elena Galán Vicente**

**Titulación: Grado de Ingeniería Mecánica**

**Profesor: Wilfried Coenen**

**Departamento: Ingeniería Térmica y de Fluidos**

**Fecha: Junio, 2013**

# Agradecimientos

A mí tutor, Wilfried Coenen, por su entera dedicación e implicación en este proyecto.

En especial, agradezco a mi familia el haberme enseñado que la lucha nunca es en vano.

A la universidad Carlos III y sus instalaciones porque sin ellas, no hubiera sido posible realizar los experimentos. Y en concreto, a los técnicos del laboratorio, por llevar a la realidad simples ideas.

# Resumen

En el presente proyecto, se analiza el flujo oscilatorio alrededor de dos cilindros. Partiendo de las bases matemáticas planteadas por **Coenen y Riley** (2009), director de este proyecto, se estudia el efecto *steady streaming* inducido por el movimiento de las paredes sólidas de los cilindros. Se pretende validar sus teorías mediante experimentos en el laboratorio del Departamento de Ingeniería Térmica y de Fluidos de la Universidad Carlos III de Madrid así como mediante simulaciones por ordenador.

Se ha observado la aparición de dos puntos de remanso y dos puntos de eyección en cada cilindro, así como, la formación de chorros que emergen desde estos puntos de eyección, formándose, a su vez, un patrón de zonas de recirculación alrededor de los cilindros.

El equipo experimental utilizado se compone de un tanque con agua mezclada con partículas de vidrío, un láser y una cámara conectada a un ordenador. Se han captado imágenes de la evolución del flujo que después han sido procesadas mediante Matlab. Así, se han obtenido las sendas de las partículas de vidrio, los campos de vectores de velocidad e isolíenas de velocidad.

Las simulaciones computacionales se han llevado a cabo mediante dos programas de CFD (*Computational Fluid Dynamics*), Fluent y Gerris. Para ello, se ha creado la geometría a estudiar y se ha generado un mallado de esa geometría. Después, se ha simulado el flujo obteniéndose las sendas correspondientes. Estos resultados se han comparado con los obtenidos experimentalmente.

Se concluye que las teorías de **Coenen y Riley** (2009) son ciertas a excepción de la hipótesis sobre la dirección del chorro formado.

# Abstract

In this project, the oscillatory flow around two cylinders has been analyzed. The s*teady streaming* phenomenon induced by the movement of solid walls of the cylinders has been studied, using as a starting pint the mathematical work from **Coenen y Riley** (2009), tutor of this project. The purpose of the present work is to validate their theory recreating the problem in the laboratory of the Thermal and Fluid Department of the University Carlos III de Madrid, and also, using computational fluid dynamics software.

The appearance of two points of attachment and two points of ejection has been observed, as well as the emergence of jets. These jets create a pattern of recirculation zones around the cylinders.

The experimental equipment consisted in a tank of water mixed with glass particles, a laser and a camera connected to a computer. Images of the evolution of the flow were taken and later were processed with Matlab. Thereby, streak lines and velocity flow fields were obtained.

The computer simulations were performed with the CFD (*Computational Fluid Dynamics*) codes Fluent and Gerris. For this, it has been necessary to create the studied geometry and to generate a mesh for this geometry. Then, the flow was solved numerically and the streak lines have been obtained. All the results have been compared with the ones obtained experimentally.

It can be concluded that all the theory raised by **Coenen y Riley** (2009) has been found in agreement with the experimental and numerical results of this work, except for the exact direction of the jets.

# Contenido

# Índice de imágenes

# Índice de tablas

# 1. INTRODUCCIÓN

# 1. INTRODUCCIÓN

## 1.1. Flujo oscilatorio alrededor de dos cilindros

En este proyecto se va a estudiar un flujo oscilatorio alrededor de dos cilindros. El movimiento de este flujo es un caso concreto de *steady streaming*. *Steady streaming* es un término inglés cuyo significado podría traducirse como 'flujo estacionario'. Existen varias formas de que se produzca el fenómeno *steady streaming*. El caso que se estudia en este proyecto es el de *steady streaming* mecánico. Éste, se define como un movimiento lento estacionario del fluido, inducido por el movimiento oscilatorio de un sólido. Este tipo de fenómeno se caracteriza por que la media temporal del campo de velocidades alrededor del cilindro es no nula.

La geometría a estudiar se compone de una pareja de cilindros simétricos de radio **a**. La distancia que separa un cilindro de otro se ha denominado, $\mathbf{g_a}$**.** Esta distancia debe ser lo suficientemente pequeña como para que los efectos de ambos cilindros interfieran entre sí (en este estudio, $\mathbf{g_a = 2a}$. Los cilindros oscilan con una amplitud **A** y una frecuencia de movimiento $\boldsymbol{\omega}$. Ver imagen 1.

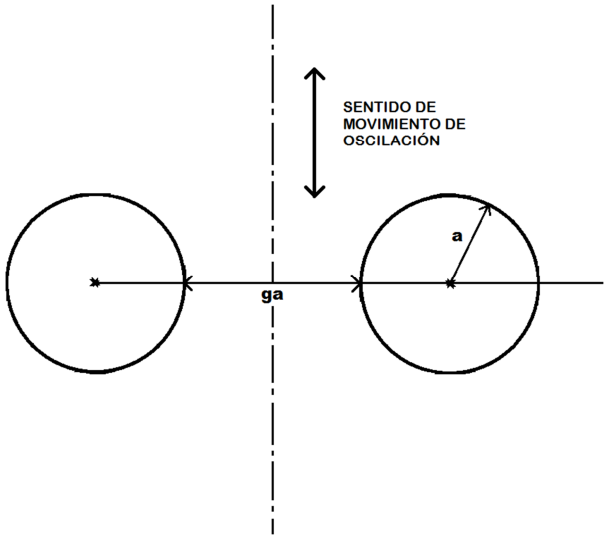



**Imagen 1. Esquema de la geometría.**

Como se explicarán más en detalle en el capítulo 2, este tipo de flujos se pueden caracterizar mediante dos parámetros adimensionales: $\boldsymbol{\epsilon = \frac{A}{a}}$ y $\mathbf{Rs} = \boldsymbol{\epsilon}\,\mathbf{Re} = \frac{\mathbf{U^2}}{\boldsymbol{\omega\upsilon}}$. Donde **Re** es el número de Reynolds y $\boldsymbol{\upsilon}$ es la viscosidad del fluido.

En el caso de este estudio se cumple que: $\mathbf{Rs} \gg \mathbf{1}$ y $\boldsymbol{\epsilon} \ll \mathbf{1}$.

Bajo estas condiciones, el flujo se comporta de la siguiente manera. Inducido por el movimiento oscilatorio de los cilindros, el fluido cerca de la pared del cilindro se va desplazando por las paredes del cilindro, aunque muy lentamente, siguiendo un movimiento

de vaivén como se ve en la imagen 2a. Se llama steady streaming a este flujo "lento" inducido por la oscilación "rápida" del cilindro. Si sólo se tiene en cuenta el desplazamiento neto del fluido tras una oscilación, el flujo de steady streaming se puede aproximar al flujo de la figura 2b.

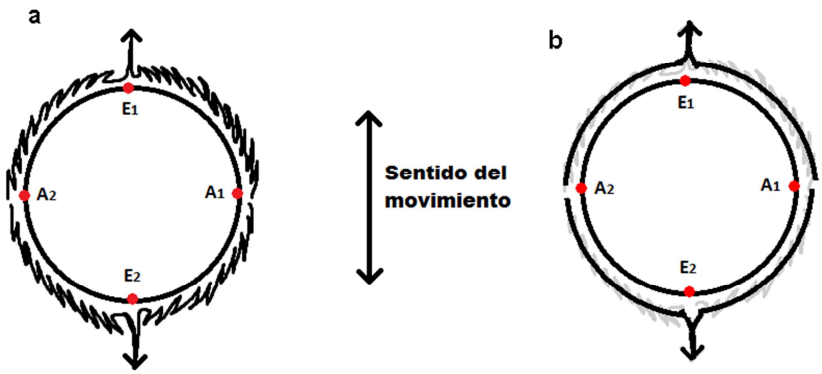



**Imagen 2. (a) Movimiento de vaivén del fluido. (b) Desplazamiento neto del fluido.**

Así, el fluido recorre una distancia, iniciando su movimiento en un punto A, llamado punto de remanso, en el que la velocidad es cero. Debido a la geometría del cilindro, existirán dos puntos A ($A_1$ y $A_2$), desde los cuales el fluido se desplazará simétricamente hasta encontrarse en un punto E, llamado punto de eyección. En este punto se producirá una colisión de ambos movimientos y el fluido será empujado hacia el exterior de las paredes del cilindro, formándose así un chorro. A todo el recorrido que realiza el fluido se le denomina senda.

El chorro de un cilindro se verá afectado por el chorro del cilindro opuesto, produciéndose un patrón de zonas de recirculación alrededor de los cilindros.

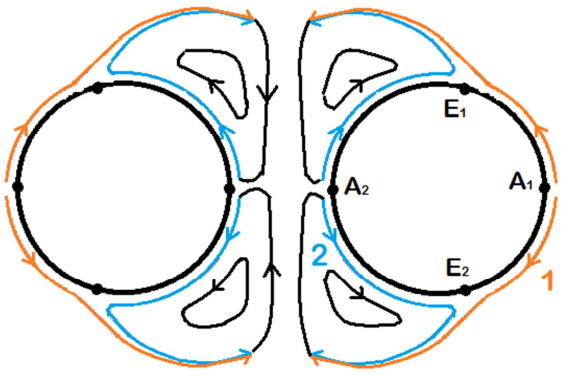



**Imagen 3. Zonas de recirculación inducidos por los chorros.**

Para el estudio de este caso de *steady streaming* se han imitado las condiciones descritas anteriormente en un laboratorio y se han obtenido fotos. Estas imágenes se analizarán utilizando Matlab donde se obtendrán las sendas y también se utilizará un método de análisis llamado PIV (*Particle Image Velocimetry)* que se explicará en detalle más adelante. Además, para respaldar los resultados se han realizado simulaciones por ordenador mediante dos programas diferentes, Fluent y Gerris y con ellos se obtendrán también las sendas.

En todos los casos se han realizado dos experimentos a y b. Pero además, en el laboratorio se ha repetido cada experimento dos veces, para tener resultados más precisos. A continuación se adjunta una tabla a modo de resumen.

**Tabla 1. Valores de los parámetros.**

| **Experimentos** | **Rs (-)** | **ϵ (-)** | **a (cm)** | **$g_a$ (cm)** |
|---|---|---|---|---|
| $a_1$ y $a_2$ | 70 | 0.2 | 0.01 | 2·a |
| $b_1$ y $b_2$ | 90 | 0.2 | 0.01 | 2·a |

Al final del proyecto se realizará una comparación de todos los resultados y se extraerán las conclusiones pertinentes.

## 1.2. Trabajos Previos

Tal y como se ha comentado al inicio el fenómeno de *steady streaming* puede venir provocado de diferentes maneras. Por ejemplo, puede estar inducido por ondas acústicas o, como en este proyecto, por un movimiento mecánico.

Cuando se habla de *streaming* acústico se refiere a un tipo de *steady streaming* provocado por ondas acústicas de alta frecuencia. Este fenómeno ha sido estudiado a lo largo de muchos años y en muy diversas aplicaciones. **Nyborg** (1953) encontró una relación entre la velocidad de un fluido sometido a s*teady streaming* y el coeficiente de absorción de un material. Sus estudios dieron pie a otros investigadores como **Piercy y Lamb** (1954). Una aplicación directa del *streaming* acústico es la levitación acústica. La levitación acústica permite mantener pequeñas partículas sólidas suspendidas en un fluido, sin necesidad de utilizar un elemento contenedor. Se consigue haciendo incidir ondas acústicas a muy alta frecuencia sobre el objeto que se desea elevar. **Chung yTrinh** (1998) usaron esta técnica para hacer crecer cristales de proteínas de manera controlada.

Para terminar, el capítulo se va a centrar en investigaciones relacionadas con s*treaming* provocado por el movimiento de una pared sólida. Y se detallarán aquellos métodos experimentales que se asemejan al implementado en este proyecto.

**Davidson y Riley** (1972) estudiaron la aparición de flujo en forma de chorro que aparecía bajo números de Reynolds altos. Un cilindro se sumergió en un tanque de agua. Éste vibraba al estar unido mediante un brazo a un vibrador. Un cable fino se colocó cerca del cilindro y a través de él se hizo pasar corriente eléctrica. Cuando el chorro formado pasaba por delante de

este cable se liberaban burbujas de hidrógeno. Estas burbujas permitían observar la estructura del chorro.

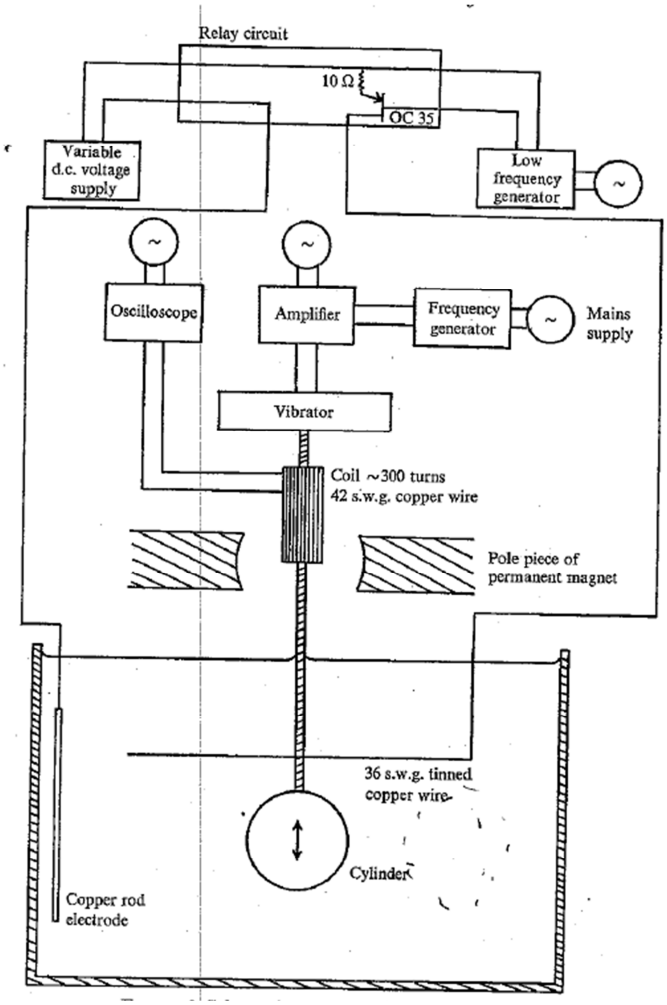



**Imagen 4. Representación esquemática del experimento de Davidson, Riley (1972).**

El cilindro usado tenía 2cm de radio. El valor de $\epsilon$ era muy pequeño, 0.05 y los valores de Rs usados eran altos, aproximadamente 300.

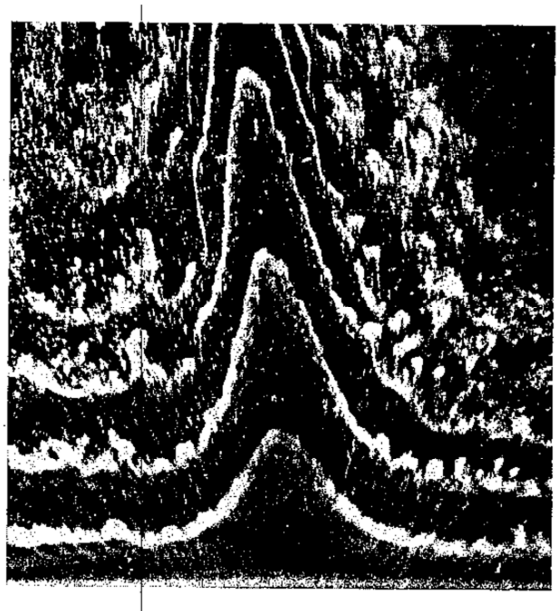


**Imagen 5. Estructura del chorro de un cilindro circular. Davidson, Riley (1972).**

Otros experimentos fueron llevados a cabo por **Wybrow & Riley** (1995). Inicialmente, analizaron el efecto *steady streaming* y el chorro formado, mediante un cilindro elíptico colocado en el interior de un tanque lleno de agua. Al igual que en el caso anterior, el cilindro vibraba con un movimiento oscilatorio. En este caso sus experimentos se centraron en el análisis de *steady streaming* inducido por oscilaciones de pequeña amplitud. El valor de $\epsilon$ fue 0.25 y $\mathbf{Rs} = \mathbf{352}$ además $\mathbf{f} = \mathbf{4Hz}$. Como se ve, el rango de trabajo de este experimento es el mismo que el del implementado en este proyecto. El agua se mezcló con partículas reflectoras de luz. Se analizaron tres tipos de movimientos, el primero desplazaba el cilindro desde el punto central de su eje. En el segundo y tercer movimientos, el brazo que movía al cilindro lo hacía desde el punto uno o dos, que estaban desviados del centro 1 o 2 cm respectivamente. Estos puntos pueden observarse en la imagen 6.

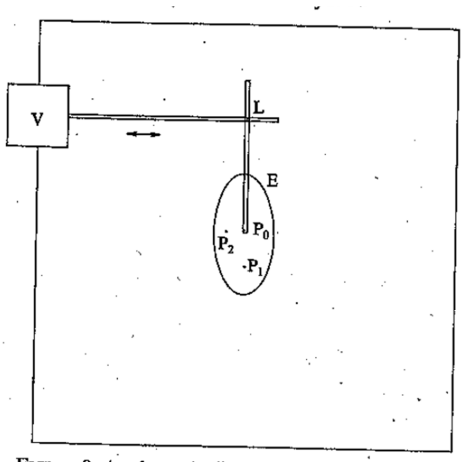



**Imagen 6. Esquema del experimento de Wybrow, Riley (1995).**

Para la iluminación se utilizaron dos proyectores que emitían una fina lámina de luz hacia la mitad del cilindro. Esta es una de las imágenes que obtuvieron.

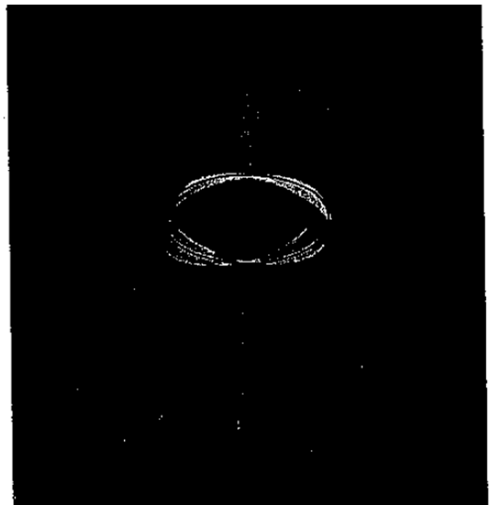


**Imagen 7. Ejemplo del inicio del chorro.**

Al año siguiente (Wybrow, Yan, & Riley, 1996), sus experimentos se basaron en simular el comportamiento del fluido cuando se colocaba un cilindro circular de 20mm de radio, cerca de una pared plana infinita. El experimento consistía en colocar un cilindro cuyo eje axial era paralelo al fondo de tanque lleno de agua, unido por cada extremo a las paredes de este, de tal modo, que la altura respecto al suelo podía variar. Un brazo unía el tanque con un disco. El brazo se atornillaba con una excentricidad al disco, así cuando el disco giraba, movido por un motor, el tanque se movía de un lado a otro, generando olas en la superficie.

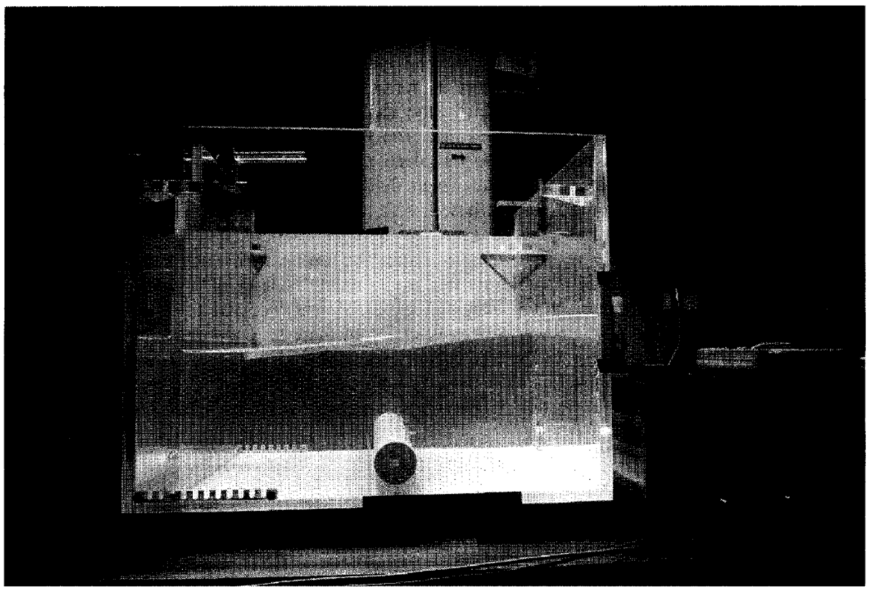


**Imagen 8. Experimento de Wybrow, Riley (1996).**

Los parámetros usados en este experimento fueron $\boldsymbol{\epsilon \approx 0.35}$ y $\boldsymbol{Rs \approx 200}$. Llegaron a la conclusión de que las partículas recirculaban desde debajo del cilindro para después ser expulsadas hacia fuera. Modelaron esta situación para demostrar que cuando existen olas que se propagan en una superficie libre, bajo una profundidad determinada, el efecto *steady streaming* puede causar el transporte de sedimentos si el lecho es móvil. Esto podría dar lugar

a efectos como el enterramiento de oleoductos o gaseoductos que pasan por debajo del mar o ser el causante de movimientos litorales de bancos de arena.

Otra aplicación relevante es la implementada por **Kotas et al** (2007). Los oídos de algunos peces (teleósteos) son capaces de identificar variaciones de presión en cualquier dirección. Sus oídos están colocados a continuación del cerebro, sin conexión ninguna con el agua del exterior. Cada oído está formado por tres órganos otolíticos, de los cuales, cada uno contiene un hueso otolito, situado dentro de una bolsa llena de líquido, que está colocada encima de una membrana sensorial. Ésta está compuesta por células capilares de las que salen pequeños capilares. Cuando se produce un sonido, el otolito oscila en el interior de la bolsa de líquido, transmitiendo esfuerzos a los capilares que se doblarán. El movimiento de estos capilares es lo que les permite a esta clase de peces determinar la frecuencia, amplitud y dirección del sonido acuático. La forma de estos otolitos se asimila a un esferoide irregular cuya dimensión total no supera un centímetro.

Para intentar comprender el funcionamiento de estos "oídos" introdujeron un esferoide en un tanque lleno de agua. El movimiento del esferoide era oscilatorio y vibraba a frecuencias de 2Hz hasta 15 Hz y la amplitud podía variar desde 0.05cm hasta 0.3cm. El fluido de trabajo fue agua (10%) y glicerina (90%), en la que se introdujeron partículas de vidrio. Una tapa de poliestireno flotaba en la parte superior del tanque para amortiguar las ondas y minimizar la evaporación de fluido. Dos focos alumbraban el tanque y en medio se colocó una cámara para captar los movimientos de las partículas de vidrio. Tal y como se ve en la imagen 9.

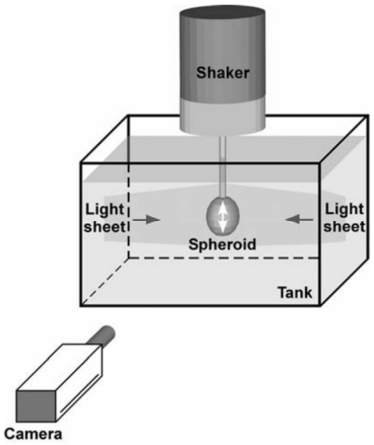



**Imagen 9. Esquema del experimento de Kotas et al (2007).**

Las imágenes captadas se analizaron mediante el método PIV. Y también se obtuvieron las sendas de las partículas de vidrio.
Los rangos de los parámetros de *streaming* que usaron fueron: $\boldsymbol{\epsilon = 0.04 - 1.2}$ y $\boldsymbol{Rs = 0.08 - 180}$.

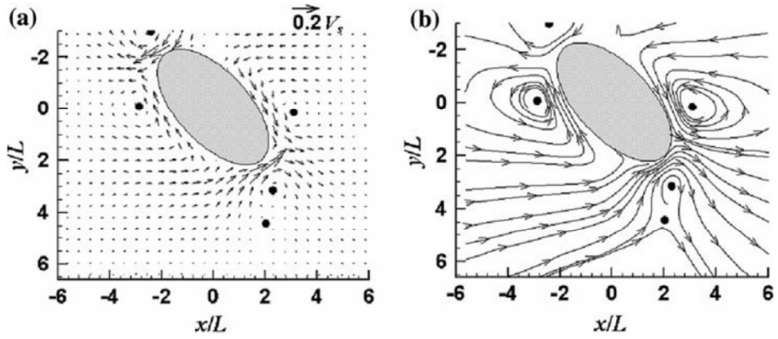



**Imagen 10. (a) Vectores resultantes del PIV. (b) Sendas resultantes. Kotas et al (2007)**

**Coenen y Riley** (2009), realizaron un análisis matemático del fenómeno s*teady streaming* alrededor de dos cilindros iguales. De hecho, este proyecto está basado en su investigación. Y se pretenden validar sus teorías. Ha de remarcarse que en ambos casos todos los análisis cumplían los siguientes requisitos fundamentales: $\mathbf{Rs} \gg \mathbf{1}$ y $\boldsymbol{\epsilon} \ll \mathbf{1}$.

De otro modo, no podrían compararse los resultados, pues, como ya se mencionó, estos parámetros definen que tipo de *steady streaming* se está produciendo.

La geometría usada en su investigación es la misma que la usada en el presente estudio.

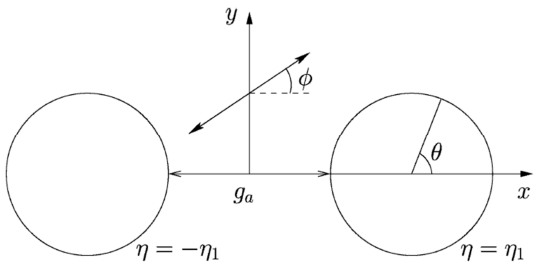



**Imagen 11. Esquema de la geometría usada por Coenen y Riley (2009).**

A diferencia de esta investigación, **Coenen y Riley** (2009) investigan el efecto *steady streaming* para distintos valores del ángulo **Φ.** Se recuerda que en este proyecto el valor de **Φ** es $\pi/2$.

Mediante recursos matemáticos aproximan las ecuaciones de Navier-Stokes para distintos valores de **Φ.**

Además fueron ellos, quienes plantearon la idea de que existieran los puntos de remanso y eyección ya mencionados al inicio. Y, plantearon que en el punto de eyección, uno de los dos flujos tendría un momento mayor debido a la influencia de un cilindro respecto del otro.

Mirando la imagen 12, se puede ver a qué momento se le ha designado el subíndice 1 y a cual el 2.

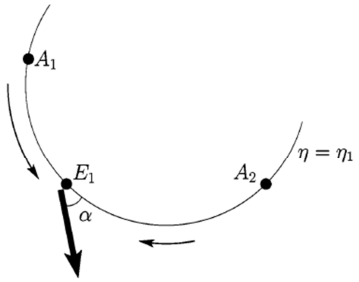



**Imagen 12. Formación del chorro en el punto de eyección. Coenen y Riley (2009).**

También, se observa que se ha llamado **α** al ángulo que forma el chorro respecto de la pared del cilindro.

Por último, formularon la hipótesis de que el momento que llevará el flujo 2 (interior) sería mayor al que tendría el flujo 1 en el punto de eyección. Por ello, predijeron que el sentido del chorro sería en dirección contraria al otro eje de simetría de los cilindros (es decir, que el ángulo α sería menor de 90 grados). La siguiente imagen muestra la predicción de **Coenen y Riley** (2009) de cómo sería el chorro para $\boldsymbol{\Phi = \pi/_2}$.

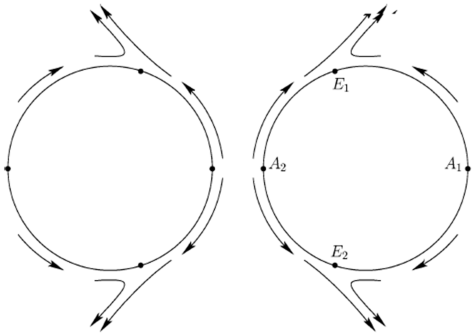



**Imagen 13. Esquema de la formación del chorro para $\Phi = \pi/_2$ . Coenen y Riley (2009).**

Como se ha expuesto, a pesar de que el flujo 2 posee mayor cantidad de movimiento, la interacción de ambos chorros hace que estos se formen hacia dentro de ambos cilindros, formándose estructuras de recirculaciones.

**Otto et al** (2008) llevaron a cabo experimentos en los que se analizaba el fenómeno *Steady Streaming* en tres dimensiones a lo largo de esferas que vibraban verticalmente. Inicialmente, experimentaron con una sola esfera, después, experimentaron con esa esfera colocada cerca del fondo de un tanque contenedor, y por último, estudiaron el flujo alrededor de dos esferas. Las esferas se sujetaban en el centro de un tanque. La geometría del tanque escogida fue octogonal ya que se asimilaba a una geometría cilíndrica pero poseía ocho paredes planas que permitían su mejor visión.

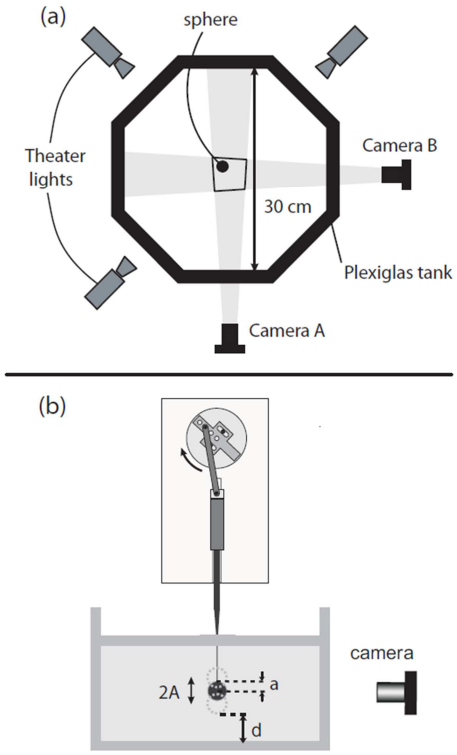



**Imagen 14. (a) Vista superior del experimento de Otto et al. (b) Vista lateral del experimento. Otto et al (2008).**

Cada esfera permanecía unida a un brazo que, como en los experimentos anteriores, estaba unido a un disco excéntrico que giraba conducido por un motor. Además, la distancia respecto del fondo podía modificarse. El tanque se rellenó con una mezcla de glicerina (92%) y agua (8%) donde la viscosidad podía ser medida constantemente gracias a un viscómetro. Partículas de vidrio reflejaban la luz de tres focos que se colocaron en tres paredes distintas para poder iluminar el tanque por completo. La frecuencia de oscilación variaba desde 1-18 Hz.

Para el análisis de imágenes usaron un método llamado 3DPTV *(3 Dimensional particle tracking velocity)* que se vale de cámaras esteroscópicas. Este método es capaz de dibujar un campo de vectores de velocidades en las tres dimensiones, de forma rápida. Para la obtención de imágenes se usaron dos cámaras de alta velocidad que captaban los movimientos de partículas reflectoras de la luz disueltas en el fluido. Para encontrar las coordenadas tridimensionales de cada partícula, hicieron coincidir la imagen de una partícula obtenida por una cámara con la imagen de la misma partícula obtenida con la otra cámara y se analizaron mediante métodos estereoscópicos. Esto es, que si se une la posición de una partícula mediante una línea recta con la cámara que tomó esa imagen y se hace lo mismo con la otra cámara estas dos líneas intersectarán en un punto del cual se conocerán las tres coordenadas espaciales.

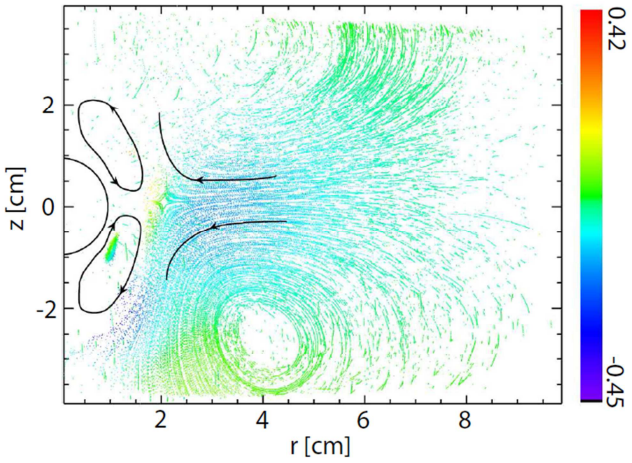



**Imagen 15. Sendas de una esfera en el centro del tanque. $\epsilon = 1.4$ y $Rs = 42$. Otto et al (2008).**

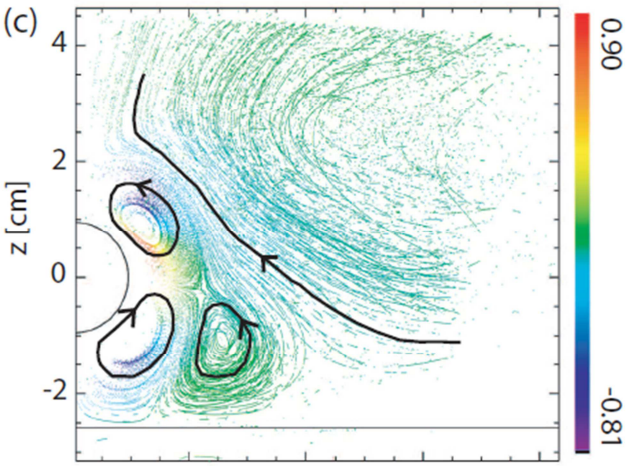



**Imagen 16. Sendas de una esfera cerca del suelo. $\epsilon = 0.7$ y $Rs = 21$. Otto et al (2008).**

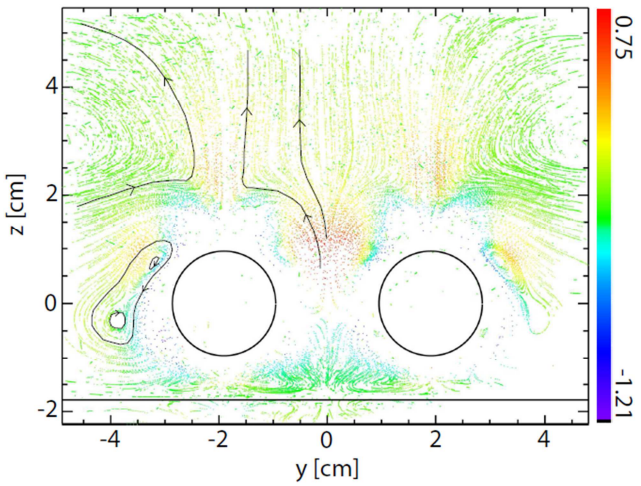



**Imagen 17. Sendas de dos esferas cerca del suelo. $\epsilon = 0.7$ y $Rs = 21$. Otto et al (2008).**

Gracias a este trabajo, encontraron un método que proporcionaba resultados precisos de un campo fluido en tres dimensiones y se explicaron ciertos comportamientos de atracción y repulsión entre las partículas fluidas.

**An et al** (2011), querían conocer los efectos que producían las olas al incidir sobre tuberías colocadas en alta mar, así como, el movimiento de sedimentos o posibles erosiones de las estructuras que se encuentran en el mar. Para ello, realizaron simulaciones por ordenador, primero en dos dimensiones, simulando la sección transversal de un cilindro circular y más tarde, simulando las tres dimensiones de esa geometría. En ambos casos las ecuaciones que utilizaron en sus simulaciones fueron las de Navier-Stokes.

Se ha de tener en cuenta que los parámetros que usaron para sus estudios fueron **KC** y **β**. A continuación, se muestran estos parámetros en función de los usados en este proyecto **Rs** y **ϵ**.

$$\mathbf{KC} = \frac{\mathbf{Um\ T}}{\mathbf{D}} = \frac{\mathbf{A\ w\ T}}{\mathbf{2a}} = \frac{\mathbf{A\ 2\pi\ f}}{\mathbf{2a}}\frac{\mathbf{1}}{\mathbf{f}} = \boldsymbol{\pi\epsilon} \quad ; \quad \boldsymbol{\beta} = \frac{\mathbf{Re}}{\mathbf{KC}} = \frac{\mathbf{Re}}{\boldsymbol{\pi\epsilon}} = \frac{\mathbf{Rs}}{\boldsymbol{\pi\epsilon^2}}$$

En cuanto al estudio en dos dimensiones (2009), la malla que usaron era una malla estructurada, con cuadriláteros y celdas más pequeñas en las zonas cercanas al cilindro.

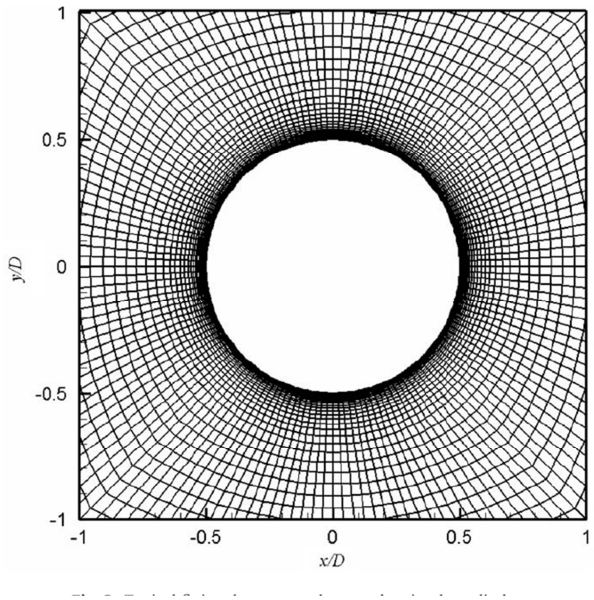



**Imagen 18. Malla usada por An et al (2009).**

En su estudio, mostraron la evolución de los vórtices según aumentaba el valor de **KC**.

En las imágenes 19 y 20 se muestran los vórtices que se forman para los mismos valores de **β** según aumenta el valor de **KC**.

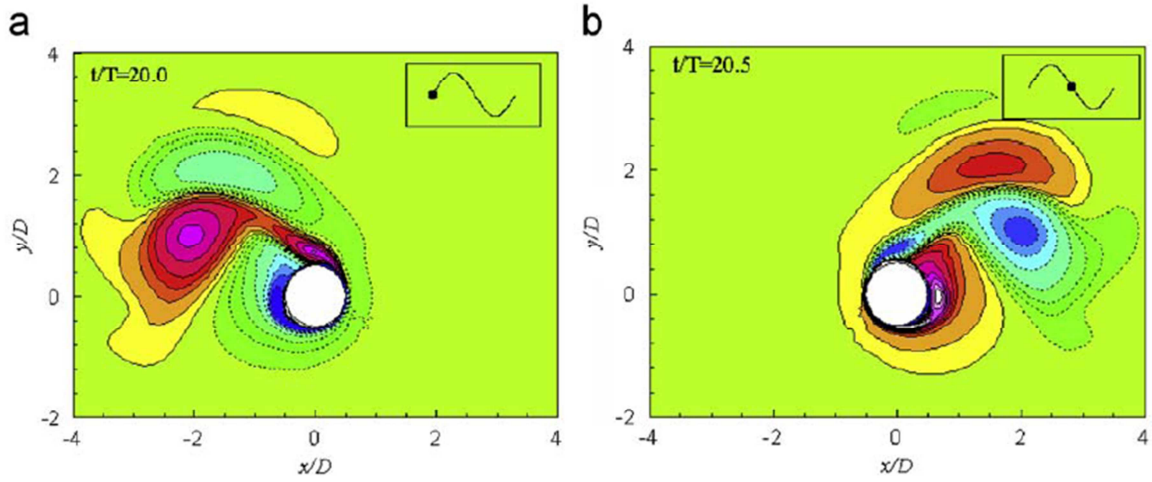



**Imagen 19. Instantánea de los vórtices para KC=10 y β=196. An et al (2009).**

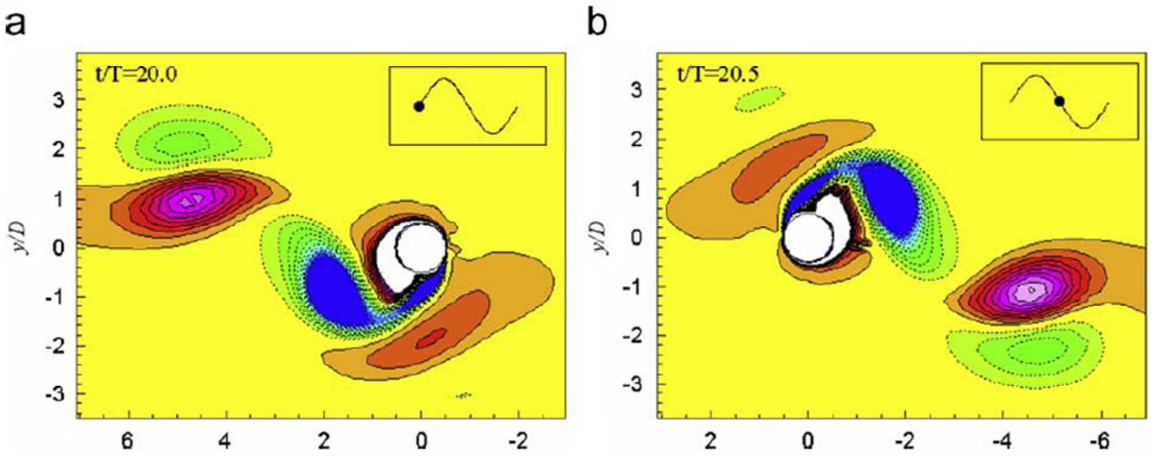



**Imagen 20. Instantánea de los vórtices para KC=16 y β=196. An et al (2009).**

Se observa, que para valores mayores de **KC** los vórtices se separan de las paredes del cilindro. Que el valor de **KC** aumente, es equivalente a decir, que aumenta el valor de **ϵ**, es decir, que la amplitud del movimiento oscilatorio aumenta.

También, muestran imágenes de cómo son las sendas en las zonas más próximas al cilindro y donde se ven las zonas de mayor velocidad de *streaming*. A modo de resumen, sólo se pondrán las imágenes que afectan a este estudio.

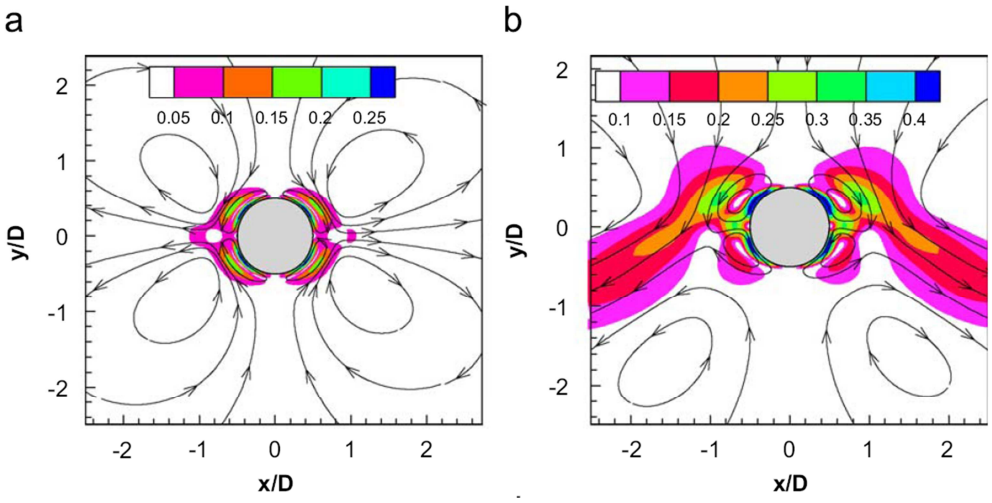



**Imagen 21. (a) Sendas cuando KC=2-4. (b) Sendas cuando KC=5-7. An et al (2009).**

Para valores de **KC** muy pequeños los vórtices quedan muy pegados a la pared del cilindro. Éste, será el caso de este proyecto, pues el valor de **KC** $\ll 1$. A medida, que el valor de **KC** aumenta estos vórtices se van haciendo cada vez más grandes hasta que se separan de la superficie sólida (**KC**=16), como se ha visto antes.

Por lo tanto, al final del proyecto, cabe esperar ver algo parecido a la imagen 21(a), aunque teniendo en cuenta que se simularán dos cilindros en vez de uno. Deberán aparecer cuatro vórtices simétricos entre los dos cilindros. Estos vórtices, no serán simétricos respecto al eje x aunque si deberían serlo respecto al y.

Para su estudio en tres dimensiones, crearon una malla como la que se muestra a continuación.

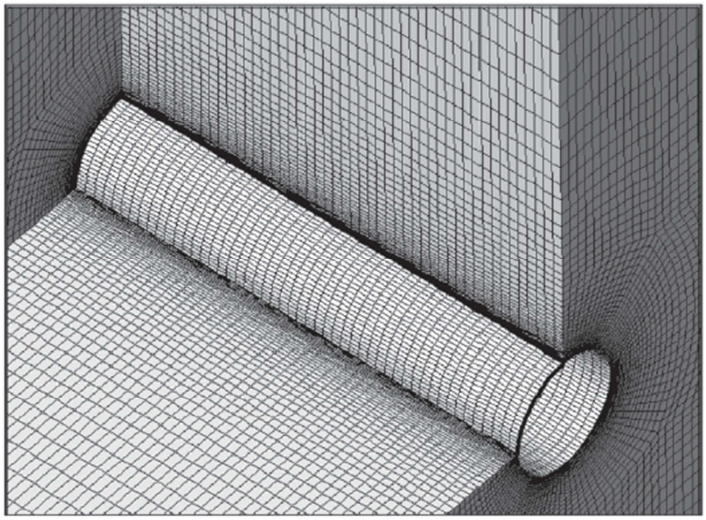


**Imagen 22. Malla en tres dimensiones de Honwei et al.**

Demostraron que los resultados eran independientes de la malla al analizar cómo cambiaba la fuerza en el eje x en cuatro mallas distintas.

Se dieron cuenta, que al analizar un flujo en tres dimensiones aparecían unos vórtices en forma de champiñón, los cuales se pueden ver en la imagen 23.

**Imagen 23. Vórtices en forma de champiñón. Honwei et al.**

Para regímenes de Re pequeños ( $\mathbf{Re} \approx \mathbf{200}$ ) el flujo apenas tenía importancia en la tercera dimensión. Sin embargo, para Re más altos ( $\mathbf{Re} > \mathbf{300}$) aparecen vórtices escalonados.

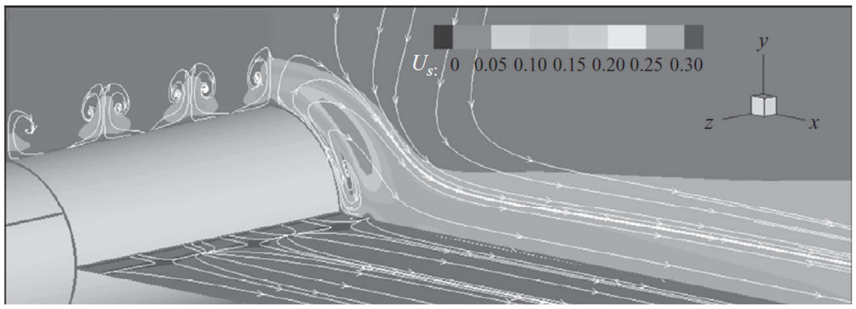



**Imagen 24. Vórtices en tres dimensiones. Para Re = 300. Honwei et al.**

**Lieu et al** (2012), usaron el efecto *steady streaming*, para la fabricación de micro dispositivos que poseían una geometría tal, que sumergidos en un fluido permitían el atrapamiento de sustancias. Crearon un método para atrapar micro partículas sin contacto alguno, de ahí, su nombre *"Hydrodynamic Tweezers"* o pinzas hidrodinámicas.
Para ello, sobre una superficie protectora de vidrio se colocaron dos discos piezoeléctricos que generaban corrientes de Eddy, o corrientes de Foucault, que impulsaban el fluido, haciendo que se moviera con movimientos oscilatorios. Éste tenía dos agujeros que permitían la entrada de fluido a un canal que estaba situado justo debajo. En el interior de este canal se colocó una geometría cilíndrica en el medio y en una de las paredes laterales un saliente y una cavidad. De tal modo, que si la sección de la pieza que se colocaba en el centro era circular en la pared se colocaría medio cilindro circular que sobresaldría de la pared así como medio cilindro circular que penetraría en la pared. Lo mismo ocurriría con las distintas secciones que se probaron en el experimento: cuadrado y diamante o rombo.

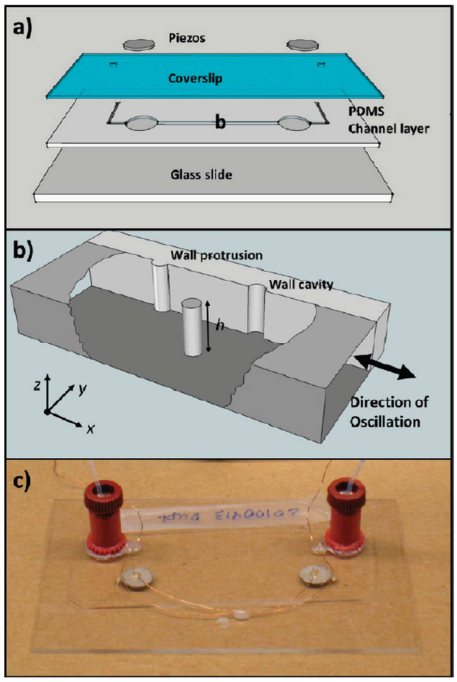



**Imagen 25.Experimento de Lieu et al (2012).**

Estas obstrucciones hacían que el fluido se desviase de su movimiento unidireccional. Se producía el efecto *steady streaming* cerca de las superficies, que provocaría recirculaciones de fluido. Descubrieron que las partículas que se encontraban en el interior de esos vórtices quedaban atrapadas permitiéndose su transporte y manipulación si necesidad de que existiera contacto.

Aunque el experimento utilizado no se parece al que se ha fabricado para este proyecto, resultan muy interesantes los resultados obtenidos en él. El valor de $\epsilon$ era mucho menor que la unidad y los valores de Re usados eran pequeños. De esto modo, se formaban recirculaciones muy cercanas a las superficies sólidas. En la siguiente imagen se muestran algunos ejemplos de estas recirculaciones.

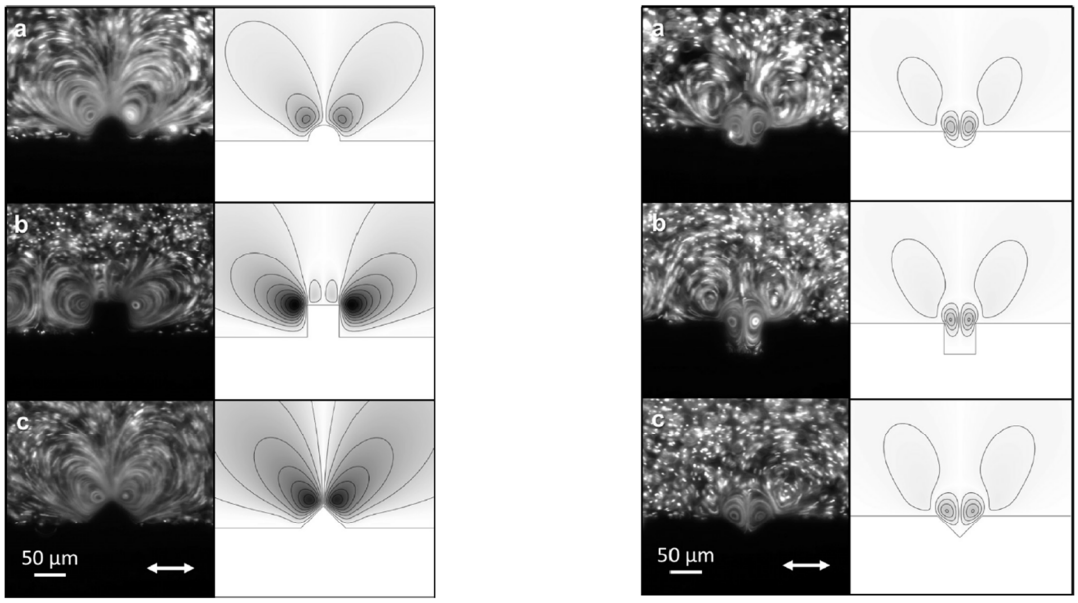



**Imagen 26. Recirculaciones en distintas geometrías. Lieu et al (2012).**

Resultados parecidos se han obtenido para este proyecto, donde las recirculaciones también se producen muy cerca de la pared del cilindro y se puede comprobar que en el interior de los vórtices una partícula quedaría atrapada.

## 1.3. Objetivos

El objetivo general de este proyecto es validar la teorías planteadas por Coenen y Riley (2009). Para ello se realizarán los siguientes pasos:

1. Estudio de las bases matemáticas de Coenen y Riley.
2. Puesta a punto del equipo experimental.
3. Diseño de casos de estudio.
4. Realización de experimentos en el laboratorio, captura de imágenes.
5. Procesamiento de las imágenes mediante Matlab.
6. Creación del caso a simular por Fluent: creación de la geometría, generación de la malla, configuración de los parámetros de la simulación, escritura de la UDF (Función de Usuario para definir la condición de contorno dependiente del tiempo).
7. Ejecución en el *cluster* de la simulación.
8. Análisis de resultados mediante el programa Tecplot.
9. Creación del caso a simular por Gerris. Codificación de funciones.
10. Ejecución en el *cluster* de la simulación.
11. Análisis de resultados con Gerris.
12. Comparación de resultados experimentales frente a los computacionales.
13. Obtención de conclusiones.

# 2. FUNDAMENTOS TEÓRICOS

# 2. FUNDAMENTOS TEÓRICOS

## 2.1. Geometría

Tal y como se indicó al inicio, se va a analizar el flujo alrededor de una pareja de cilindros idénticos entre sí. Poseen un radio $\mathbf{a}$ y están separados entre sí por una distancia $\mathbf{g_a}$, que será fija durante todo el proyecto, $2\mathbf{a}$. A continuación se muestra un esquema de la geometría.

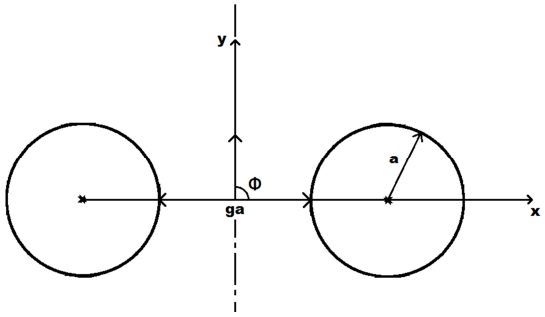



**Imagen 27. Parámetros de la geometría.**

El ángulo $\boldsymbol{\phi}$ se forma de la intersección del eje horizontal con la dirección del movimiento de los cilindros, $\boldsymbol{\phi = \pi/2}$, es decir, el movimiento de los cilindros tendrá el sentido del eje **y**. Nótese, que el mismo efecto se conseguiría si en vez de mover los cilindros se moviese el fluido. Así lo hicieron en sus experimentos **Wybrow, Riley** (1996). Se ha escogido mover los cilindros por sencillez en el mecanismo del experimento.

Los cilindros se mueven siguiendo un movimiento oscilatorio, que se define a continuación.

## 2.2. Movimiento oscilatorio de los cilindros

La posición del centro del cilindro se expresa de la siguiente manera.

[1] $$\mathbf{x(t) = Acos(\omega t + \varphi)}$$

Donde A es la amplitud de oscilación, w es la velocidad angular, t es el tiempo y φ representa la fase inicial que indica el estado de oscilación en el instante inicial (t=0). A modo de simplificación se ha considerado $\boldsymbol{\varphi = 0}$.

La velocidad del centro del cilindro es la derivada de la posición respecto del tiempo.

[2] $$\mathbf{V(t) = -\omega Asen(\omega t) = Usen(\omega t)}$$

Donde **U** es la velocidad máxima: $\mathbf{U = A\omega}$

Por otro lado, la frecuencia de oscilación se puede escribir como:

[3] $$\mathbf{f = \frac{\omega}{2\pi}}$$

## 2.3. Navier-Stokes

Las ecuaciones matemáticas que describen el fenómeno *steady streaming* son las ecuaciones de *Navier-Stokes,* ya que estas describen el movimiento de toda partícula fluida. Éstas, se muestra a continuación.

[4] $$\boldsymbol{\nabla \cdot \mathbf{v} = 0}$$

[5] $$\boldsymbol{\frac{\partial \vec{v}}{\partial t} + \vec{v} \cdot \nabla \vec{v} = -\frac{1}{\rho}\nabla p + \upsilon \nabla^2 \vec{v}}$$

Donde **v** es la velocidad del fluido **ρ**, es la densidad, **p** es la presión y **υ** es la viscosidad cinemática.

Los términos $\boldsymbol{\vec{v} \cdot \nabla \vec{v}}$ representan el transporte convectivo mientras que $\boldsymbol{\upsilon \nabla^2 \vec{v}}$ representa el transporte por difusión viscosa.

Para adimensionalizar las ecuaciones de Navier-Stokes, hay que dividir por los parámetros característicos que correspondan. Se usa como velocidad característica, **U**, como tiempo característico, $\boldsymbol{1/\omega}$ y como longitud característica **a**. Cuando esta ecuación se adimensionaliza, aparecen dos parámetros fundamentales en el estudio de s*teady streaming.* Estos parámetros son **ϵ** y **Rs**.

El primero, representa la relación entre la amplitud de oscilación del movimiento oscilatorio respecto del radio del cilindro.

[6] $$\boldsymbol{\epsilon = \frac{A}{a} = \frac{U}{\omega a}}$$

El otro parámetro a definir es **Rs** o *Streaming Reynolds number*. Que se define como el número de Reynolds multiplicado por $\boldsymbol{\epsilon}$.

[7] $$\boldsymbol{Rs = \epsilon Re = \epsilon \frac{aU}{v} = \frac{U^2}{\omega v} = \frac{\epsilon^2 a^2 2\pi f}{v}}$$

Así, la frecuencia de movimiento queda relacionada con el valor de **Rs**.

La resolución analítica de la ecuación de Navier-Stokes para este flujo resulta imposible. Sin embargo, mediante las técnica matemática de expansiones asintóticas, **Coenen y Riley** (2009) obtuvieron soluciones para la ecuación en el límite $\boldsymbol{Rs \gg 1}$ y $\boldsymbol{\epsilon \ll 1}$.

Para entender qué significa un valor de **ϵ** muy pequeño, se recurre a otro número, el número de Strouhal (St). Éste, representa los efectos no estacionarios frente a la inercia de la partícula fluida.

[8] $$\boldsymbol{St = \frac{\omega a}{U}}$$

Tal y como se observa, el número St es el inverso del parámetro **ϵ**. Así pues, cuando **ϵ** es mucho menor que 1 significa que los efectos no estacionarios dominan sobre los efectos de inercia del flujo.

[9] $$\mathbf{St} = \frac{\omega a}{U} \gg 1 \quad \leftrightarrow \quad \epsilon = (\mathbf{St})^{-1} \ll 1$$

Desde un punto de vista experimental, indica que la amplitud del movimiento oscilatorio es pequeña comparada con el radio del cilindro.

Ha de tenerse en cuenta que el número de **Re** es mucho más grande que el número **Rs** si $\epsilon$ es menor que 1. Esto es importante, pues a pesar de que **Rs** debe ser mucho mayor que la unidad no debe excederse de un cierto valor pues haría que el flujo fuese turbulento, creándose un tipo de flujo que no es el que aquí se desea estudiar.

En este caso de s*treaming*: $\mathbf{Rs} \gg \mathbf{1}$ y $\epsilon \ll \mathbf{1}$, el flujo se puede dividir tres zonas características, que se definen a continuación.

## 2.4. Zonas en Steady Streaming

La primera de ellas, es conocida como la capa de Stokes. El espesor de esta capa es el siguiente.

[10] $$\mathbf{St_{espesor}} = \frac{\epsilon}{\sqrt{Rs}}$$

Esta capa es la más cercana a la superficie del cilindro. Este espesor es realmente pequeño, si se tiene en cuenta que $\epsilon$ es muy pequeño y que Rs es muy grande. En esta capa predominan los esfuerzos viscosos-

A continuación, se desarrolla otra capa conocida como *capa límite* o en inglés como *boundary layer*.

[11] $$\mathbf{B_{espesor}} = a\frac{\epsilon}{\sqrt{Rs}}$$

La capa límite, será **a** veces más grande que la capa de Stokes.

Resulta interesante desarrollar las ecuaciones [10] y [11], introduciendo las expresiones [8] y [9].

[12] $$\mathbf{St_{espesor}} = \frac{\epsilon}{\sqrt{Rs}} = \frac{\frac{U}{\omega a}}{\sqrt{\frac{U^2}{\omega \nu}}} = \frac{\sqrt{\nu}}{\sqrt{\omega}} \cdot \frac{1}{a}$$

Del mismo modo:

[13] $$\mathbf{B_{espesor}} = \epsilon\frac{a}{\sqrt{Rs}} = \frac{\sqrt{\nu}}{\sqrt{\omega}}$$

Se puede observar, a primera vista, que existe una diferencia de espesor entre las capas de $\mathbf{1}/\boldsymbol{a}$.

El espesor de la capa queda definido cuando se relaciona la viscosidad con la frecuencia de vibración. Resulta fácil de entender, que cuanto más rápido vibre un objeto, menos espesor

poseerá tanto su capa de Stokes como su capa límite. O dicho de otro modo, cuanto más viscoso sea el fluido, mayor serán los espesores de estas capas.

Es decir, que esta capa tiene una relación directa con la tendencia de un fluido a quedarse "pegado" a la superficie de un sólido.

A continuación de la *capa límite,* la zona más alejada de la superficie, el fluido se rige por las ecuaciones del movimiento oscilatorio [2], explicado anteriormente. En esta zona el fluido se puede considerar no viscoso.

Ahora, se explicarán con detalle los sucesos que ocurren en cada capa y por ende, cómo se produce el fenómeno s*teady streaming.*

Como se ha explicado anteriormente, en la zona más próxima a la superficie los efectos viscosos son más importantes, por lo tanto, estos se concentran en la capa de Stokes. El resto de zonas pueden considerarse como no viscosas. Lo que significa que los esfuerzos viscosos sólo se producen muy cerca de las superficies de los cilindros. Se puede demostrar analizando la ecuación de Navier-Stokes aplicada a esta zona, en la que el transporte por difusión (donde aparecen los términos viscosos), resulta ser un término superior frente al transporte convectivo. Ver ecuación [5].

Otra manera, quizá, más sencilla de demostrar la predominancia de la viscosidad en esta zona, es aplicando un número de *Reynolds* local. Este número relaciona los términos convectivos frente a los viscosos de la ecuación de Navier-Stokes.

[14] $$\mathbf{Re_{local}} = \frac{\mathbf{velocidad\ cerca\ de\ la\ superficie}}{\mathbf{v}} . \mathbf{distancia\ a\ la\ superficie}$$

Como el espesor de la capa de Stokes es muy pequeño la distancia a la superficie será casi nula y la velocidad muy cerca de la pared tiende, también, a cero. Por lo tanto el $\mathbf{Re_{local}}$ será menor que la unidad. En conclusión, la viscosidad predomina frente a los efectos dinámicos.

Gracias a esta viscosidad, el fluido queda "atrapado" en las paredes del sólido. Por otro lado, los valores de la velocidad serán fluctuantes ya que el cilindro tiene un movimiento senoidal. Por lo tanto, se tienen unas fuerzas viscosas y unas fuerzas de inercia debidas al movimiento de oscilación de los cilindros. En la capa de Stokes, los esfuerzos viscosos son más fuertes que las fuerzas de inercia por lo que el fluido en esas zonas quedará unido al movimiento que hagan las paredes del cilindro. Sin embargo, en la Capa Límite los esfuerzos viscosos son más débiles que las fuerzas de inercia y por tanto existirá un movimiento, aunque muy lento. Es decir, el movimiento *Streaming* se produce en la capa Límite, influenciado por los sucesos de la capa de Stokes.

Mirando la imagen 2, mostrada al inicio del proyecto, se observa que este movimiento es muy lento, casi estacionario, ya que el fluido tiene a seguir el movimiento del sólido.

## 2.5. Puntos de eyección y remanso

En un movimiento senoidal (ver ecuación [2]), existirán dos puntos done el seno sea igual a 0 y por tanto la velocidad sea nula. Estos puntos, se llaman puntos de remanso (**A**). Desde ellos, el fluido 'empujado' por la viscosidad de la pared del cilindro tenderá a moverse desde el punto de remanso $\mathbf{A_1}$ hasta el punto de remanso $\mathbf{A_2}$ pegados a la pared. Tal y como se observa en la imagen 28.

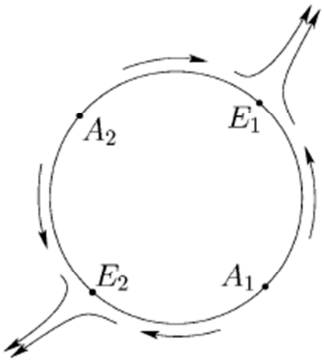



**Imagen 28. Puntos de remanso (A) y eyección (E) y formación del chorro. Imagen obtenida de Coenen y Riley.**

En fluido llegará a los puntos $\mathbf{E_1}$ y $\mathbf{E_2}$ desde direcciones contrarias. Estos puntos se conocen como puntos de eyección. Esto es así, porque en los puntos de choque el fluido es empujado en la dirección contraria a la superficie sólida formándose un chorro.

En el caso de que exista simetría la cantidad de movimiento generada en ambos lados del cilindro será igual y por tanto el chorro se producirá en el punto medio de la circunferencia y saldrá con un ángulo **α** de 90 grados. Siendo **α**, el ángulo indicado en la imagen 12 de **Coenen y Riley**. Es decir, que el chorro será perpendicular al cilindro.

La dificultad, por el contrario, reside cuando no existe simetría. Esta falta de simetría puede producirse debido a la geometría analizada o como es el caso, debido a la existencia de otro cilindro separado a una distancia suficientemente pequeña como para interferir en el primero. Lo que ocurrirá en este caso es, que el fluido llega al punto de eyección por un lado con una cantidad de movimiento $\mathbf{M_{1e}}$ y en dirección opuesta llegará con otra cantidad de movimiento $\mathbf{M_{2e}}$. Según los resultados obtenidos analíticamente, (Coenen & Riley, 2009) $\mathbf{M_{2e}} > \mathbf{M_{1e}}$ y aunque cabría esperar que el chorro tomase la dirección del vector de cantidad de movimiento $\mathbf{M_{2e}}$ por ser mayor, se va a demostrar en este trabajo que el chorro se forma hacia el interior formándose recirculaciones. Esto es debido a la interacción entre ambos chorros.

A lo largo de los siguientes capítulos, se demostrará dicha dirección de los chorros y la existencia de los puntos de eyección y remanso mediante experimentación en un laboratorio y mediante análisis computacionales.

# 3. DESARROLLO EXPERIMENTAL

# 3. DESARROLLO EXPERIMENTAL

Se ha realizado en un laboratorio un experimento que representa las condiciones explicadas en el capítulo de fundamentos teóricos. Se introducirán partículas de vidrio de la misma densidad que el agua. Estas partículas de vidrio se comportarán como lo haría el agua. Se analizará el recorrido que siguen para conocer el movimiento del fluido.

Para ello, se van a obtener unas imágenes tomadas con una cámara y después se analizarán mediante Matlab.

Al final se presentarán como resultados las sendas recorridas por el fluido y un campo de vectores de velocidad.

Los resultados que se van a comentar en este capítulo son los procedentes de cuatro experimentos. Para distinguirlos se llamarán por las letras $\mathbf{a_1}$, $\mathbf{a_2}$, $\mathbf{b_1}$ y $\mathbf{b_2}$ respectivamente. Los experimentos $\mathbf{a_1}$ y $\mathbf{a_2}$ son iguales entre sí, pero se han repetido para obtener más precisión en los resultados. Lo mismo ocurre con los experimentos $\mathbf{b_1}$ y $\mathbf{b_2}$.

Del experimento $\mathbf{a_1}$ se tomaron 568 imágenes, del $\mathbf{a_2}$ 632, del $\mathbf{b_1}$ 609 imágenes y del $\mathbf{b_2}$ 608.

A modo de resumen se incluye una tabla con los valores que son comunes en los cuatro experimentos.

**Tabla 2. Valores de los parámetros de los experimentos.**

| Variable | Valor |
|---|---|
| Amplitud de movimiento (A) | 0.002 [m] |
| Radio cilindro (a) | 0.01 [m] |
| Densidad el agua | 1000 [kg/m$^3$] |
| Viscosidad cinemática ($\boldsymbol{\nu}$) | $1 \cdot 10^{-6}$ [m$^2$/s] |
| Distancia entre cilindros (ga) | 2·a [m] |

A continuación, se explica en detalle el experimento.

## 3.1. Esquema del experimento y funcionamiento del conjunto

El funcionamiento general se describe a continuación.
Un tanque transparente de plástico se llena de agua mezclada con partículas de vidrio. Dos cilindros se colocan en el interior del tanque. Ambos cilindros están unidos por el extremo superior a una base que se mueve siguiendo un movimiento oscilatorio gracias a un motor. Para la captación de imágenes, se utiliza una cámara fotográfica. Ésta se coloca en la parte inferior del tanque. De esta manera, la cámara queda enfocando a un espejo colocado justo debajo de los cilindros, el cual está orientado a 45 grados. La cámara obtendrá imágenes de dos circunferencias (plano transversal de los cilindros) y las partículas a su alrededor. Nótese, que la intención de este proyecto es el análisis en dos dimensiones. Las partículas de vidrio reflejan la luz proporcionada por un láser colocado en un extremo del tanque. El funcionamiento se puede ver en los esquemas mostrados a continuación.

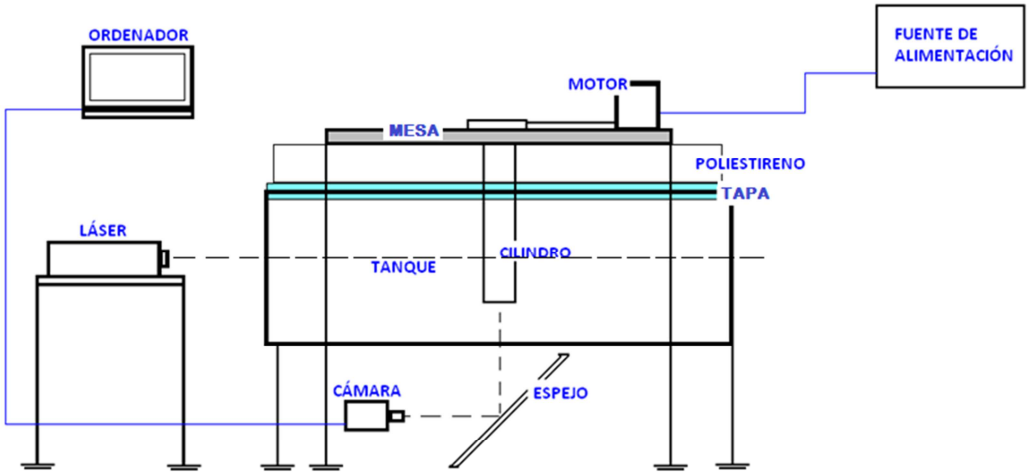



**Imagen 29. Esquema general del experimento. Alzado.**

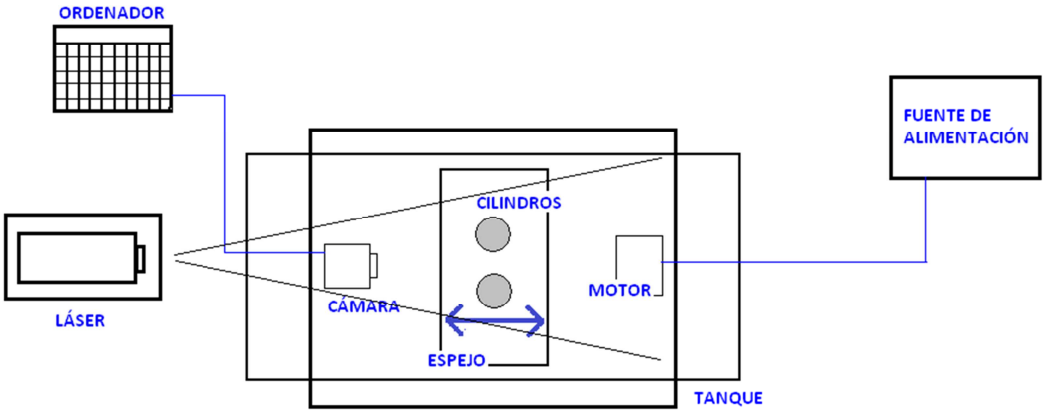



**Imagen 30. Esquema general del experimento. Planta.**

## 3.2. Detalle de cada elemento

### 3.2.1. Tanque

Se ha utilizado un tanque de 104x42x40 $cm^3$ de plástico transparente que tiene paredes de 1cm para soportar la presión del agua.

Los laterales se han cubierto de cartulina negra para evitar que la luz pase. Se ha realizado una pequeña abertura en uno de los extremos para que el haz del láser pueda penetrar al interior del tanque.

Tal y como se observa en la imagen 31, en la parte superior del tanque hay una superficie de plástico donde se apoyarán los cilindros.

**Imagen 31. Fotografía al tanque y la superficie de plástico.**

Para evitar oleaje, en la superficie se ha colocado una tapa flexible de espuma, como se ve en la imagen 32. En ella, se ha realizado un agujero en el centro para que los cilindros puedan atravesarla. Además, la tapa flexible se ha unido a unos listones de poliestireno, que quedan perfectamente acoplados a la tapa superior, gracias a la flotabilidad que ésta posee sobre el nivel de agua del tanque. El lado de la tapa de espuma que queda orientada hacia el interior del tanque se ha recubierto de una lámina de gomaespuma negra para eliminar los reflejos que se puedan producir.

**Imagen 32. Fotografía de la esterilla y el poliestireno.**

Sobre las patas del tanque se han colocado unos topes sobre los que va apoyado el espejo a 45 grados. Ver imagen 33.

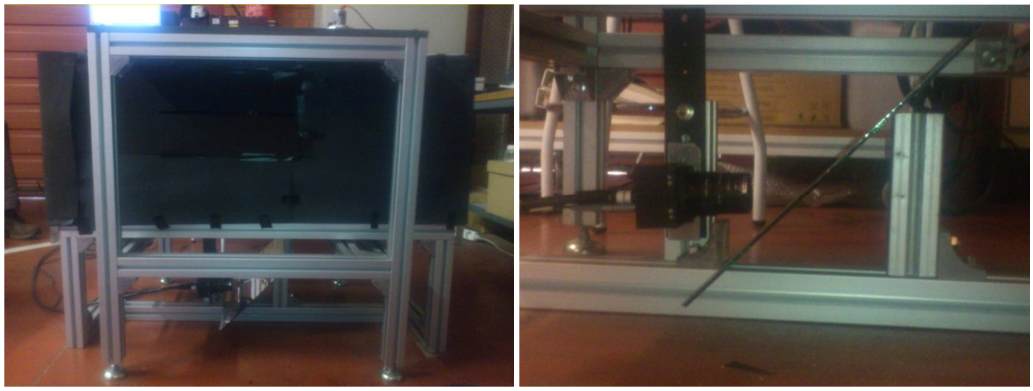


**Imagen 33. (a) Tanque completo. (b) Detalle del espejo y sus apoyos.**

### 3.2.2. Fluido

El fluido de trabajo es agua mezclada homogéneamente con partículas de vidrio de una micra de radio. Las partículas de vidrio usadas se llaman Sphericel 110P8, fabricadas por la empresa Potters-Ballotini.

### 3.2.3. Cilindros y acoplamiento

Se han utilizado dos cilindros de plástico de un centímetro de radio cada uno. Separados entre sí por una distancia de un diámetro. Para evitar que reflejen la luz se le ha aplicado una pintura negra mate.

Los cilindros se atornillan a una base semicircular metálica, cuya zona interior también se ha pintado de negro. En la imagen 34 se puede ver como la base se atornilla a un aro metálico que se ha encajado a la tapa superior. El interior de este aro está vacío para que los cilindros pasen al interior del tanque.

**Imagen 34. (a) Cilindros atornillados a su base. (b) Detalle del aro metálico.**

Este aro metálico se mueve por la tapa mediante unos raíles, y además, se ha sujetado con unos topes dorados (ver imagen 35), para forzar su movimiento en una dirección. Para mejorar su deslizamiento es necesario lubricar los raíles, los topes y el aro con vaselina cada cierto tiempo.

**Imagen 35. Topes.**

El aro metálico está unido a un brazo cilíndrico metálico que posee un codo. Ver imagen 36. El codo permite la absorción de movimientos en aquellas direcciones que no sea específicamente la longitudinal.

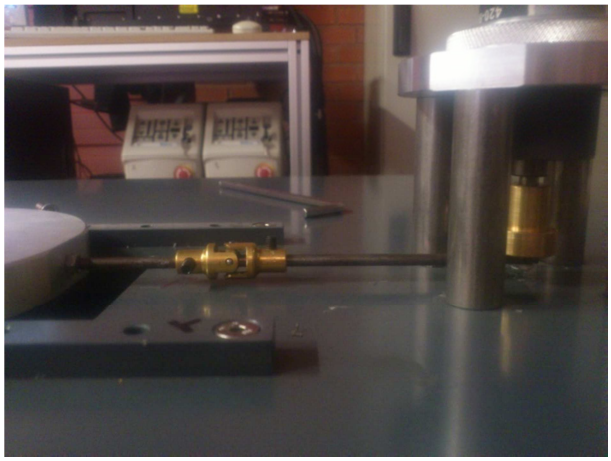


**Imagen 36. Brazo que une la base de los cilindros con el motor.**

El otro extremo del brazo se atornilla a una pieza que a su vez está unida al motor. Ésta pieza es la que transmite el movimiento del motor a la base de los cilindros, la cual por uno de sus extremos se fija al motor, y éste la hace girar, ver imagen 37. Por el otro lado se le han realizado pequeños agujeros con distintas desviaciones respecto del centro. El brazo se fijará a cualquiera de estos agujeros, según la excentricidad que se desee. De tal forma, que cuando el motor gire, también lo hará la pieza agujereada y a su vez moverá al brazo. El brazo traducirá el movimiento circular en uno longitudinal moviendo la base y con ella los cilindros, que realizarán el movimiento oscilatorio deseado.

**Imagen 37. Pieza dorada (a) En su posición de funcionamiento. (b) Agujeros desviados del centro.**

Cuanto mayor sea la excentricidad escogida, mayor será la amplitud del movimiento oscilatorio de la base. Para los casos expuestos en este experimento, la excentricidad escogida ha sido de 0.02 m. Observando la ecuación [6], del primer capítulo, significa que el valor de $\in$ utilizado es de 0.2 pues, como se ha comentado anteriormente, el radio del cilindro es de 0.01 m.

### 3.2.4. Motor

El tipo de motor utilizado proporciona un voltaje de 4.5-15V D.C. y una potencia de 21.2W. Se alimenta con una fuente regulable que trabaja desde 0 a 30 V. Se ha escogido una fuente regulable para poder trabajar a distintas frecuencias. Los experimentos se han realizado a 3-4 Hz aproximadamente.

Para la incorporación del motor, se ha construido un pequeño soporte metálico que se ha situado en la tapa superior. La fuente de alimentación se coloca en una mesa contigua.

**Imagen 38. Motor.**

**Imagen 39. Motor colocado en su soporte.**


**Imagen 40. Fuente de alimentación.**

### 3.2.5. Láser

El láser que se ha utilizado posee una longitud de onda: 300-2000mm. Emite una energía de 400mJ. Posee unas dimensiones aproximadas de 43mm de diámetro y 90mm de longitud. El material de la lente es cuarzo.

La única luz que debe entrar en el tanque ha de ser la del láser. Así, cuando se visualicen las imágenes captadas por la cámara se verá todo negro, a excepción de las minúsculas partículas de vidrio, que se verán blancas.

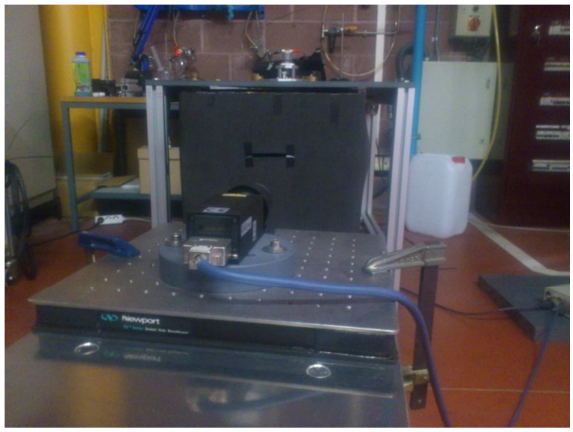


**Imagen 41. Láser y cartulina con un pequeño corte.**

El láser se apoya sobre una mesa que se ve en la imgen 40, cuyas patas son regulables. Se pretende que el haz del láser sea lo más horizontal posible e incida hacia la mitad de la longitud de los cilindros.

**Imagen 42. Láser apoyado en su mesa.**

El láser se coloca en el extremo mostrado en el esquema al inicio del capítulo. Esto es así, para poder iluminar a los dos cilindros, porque el efecto que se desea observar ocurre entre ambos. Véase el siguiente esquema.

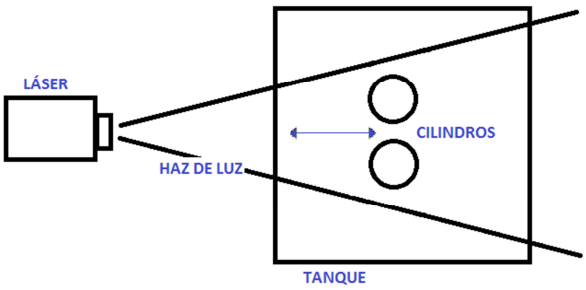



**Imagen 43. Esquema de cómo índice la luz en los cilindros.**

### 3.2.6. Cámara y software que utiliza

Para que la cámara se mantenga en la posición deseada se le ha construido un soporte que queda unido a las patas del tanque tal y como se ve en la imagen 44. Este soporte se puede alejar o acercar del espejo y subir o bajar en altura.

La cámara realiza 30 fotos por segundo y permite regular, el zoom, el diafragma (controla la exposición a la luz) y el enfoque.

La cámara se conecta mediante un cable a un ordenador. Las imágenes se ven en la pantalla gracias a un software llamado Insight 4G. Con él, se pueden crear ficheros de imágenes que se graban cuando se desee. Además, este software está pensado para la sincronización del láser con la cámara. No obstante, en este proyecto el láser siempre permanecerá encendido.

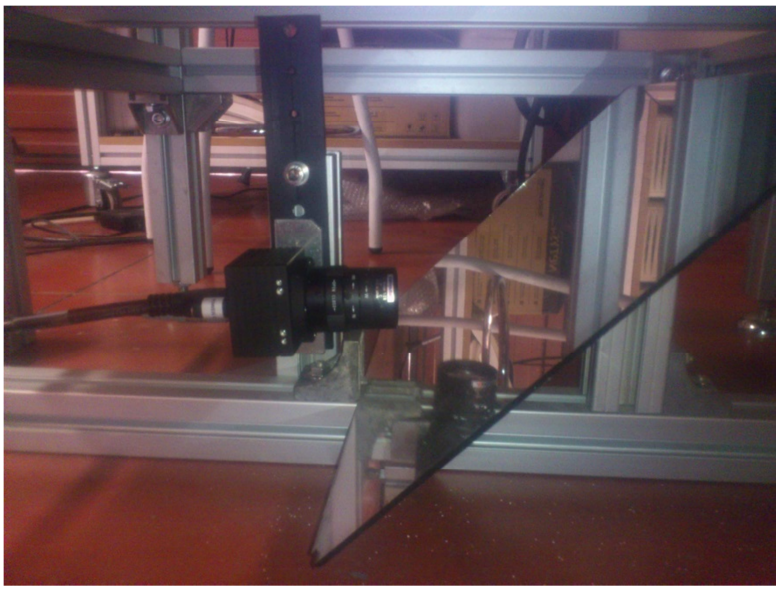


**Imagen 44. Cámara y su soporte.**

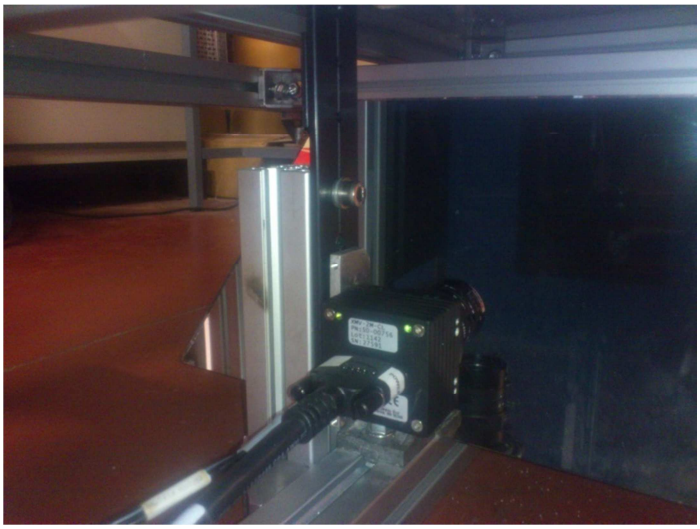


**Imagen 45. Detalle del soporte de la cámara.**

## 3.3. Análisis de las imágenes reales

Las imágenes exportadas por la cámara se obtuvieron en un formato de 12 bits que no permitía su lectura con cualquier programa de visualización de imágenes. Por ello, se utilizó un software llamado Fiji. Cuyo funcionamiento es muy simple, pero permite hacer una pre-visualización de las imágenes antes de proceder a su análisis.

Para obtener los resultados deseados a partir de esas imágenes se han realizado cuatro procesos diferentes. Para todos ellos se ha utilizado Matlab 2009.

Ha de aclararse que cada uno de los programas analiza un directorio en el que están contenidas una cantidad suficiente de imágenes como para que se consideren los resultados como válidos. (Número de imágenes superior a 500).

### 3.3.1. Frecuencia de oscilación

Debido a que no existe ningún tipo de sincronización entre la posición del cilindro y la cámara, no se conoce la frecuencia exacta de movimiento del cilindro en un experimento ni tampoco se sabe en qué posición se encuentra el cilindro cuando se tomó la foto. Por ello, se ha realizado un programa llamado '*buscar_freq.m*' que calcula el periodo. En el anexo A se adjunta el programa.

Cuando se pone en funcionamiento el programa, muestra una imagen, en la que el usuario debe escoger una región a analizar (región de interés). Esta región debe cubrir parte del cilindro así como un trozo de alrededor. Ver imagen 46.


**Imagen 46. Esquema de la región de interés.**

El programa genera una matriz, instantánea, con los valores de grises de esa zona. De tal modo, que al ser el cilindro más oscuro que el fondo, esa zona será más gris que el resto. Cuando el cilindro se mueva, los valores de grises de esa matriz se verán modificados. Mediante la transformada de Fourier se obtienen las frecuencias con la que estos valores de grises varían y así se puede conocer la frecuencia de movimiento del cilindro.

El programa muestra al final un espectro de frecuencias como la mostrada a continuación.

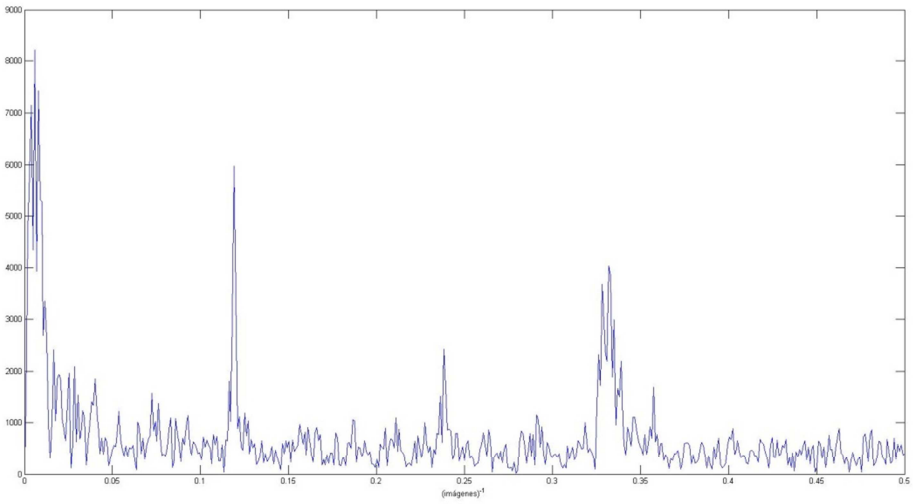



**Imagen 47. Frecuencia de oscilación.**

En esta última figura, si existe una frecuencia de movimiento clara, se verá un pico más alto que el resto. La proyección del valor de ese pico sobre el eje de abscisas será un valor numérico. La inversa de ese valor nos dará el número de imágenes que se tomaron en un ciclo. Este valor se ha llamado $\boldsymbol{\delta}$.

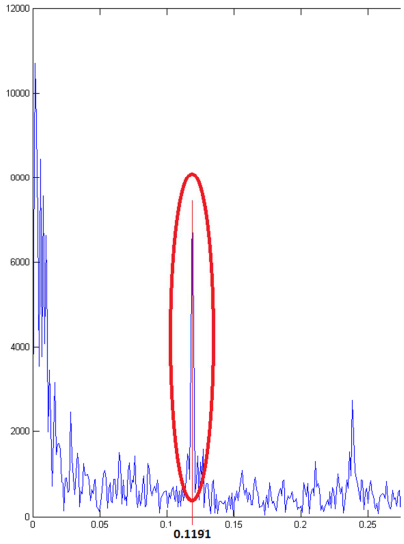



**Imagen 48. Obtención del valor $\delta$**

Como se sabe que la cámara realiza 30 fotos por segundo, para conocer la frecuencia sólo es necesario dividir 30 entre $\boldsymbol{\delta}$.

[15] $$\boldsymbol{f[\mathrm{Hz}] = \frac{30\ (imágenes/segundo)}{\delta\ (imágenes)}}$$

### 3.3.2. Sendas de las partículas

Mediante este programa, se procesan las imágenes para ver el recorrido que han seguido las partículas mediante un programa llamado '*sendas_promediado.m', v*er anexo B

Este programa aplica primero, un filtro que ajusta los valores de los grises a cada imagen. (Filtro '*imadjust'* de Matlab*).*

Por otro lado, mediante el valor entero de $\delta$, obtenido en la fase inicial, se pueden agrupar todas las imágenes en las que el cilindro esté en la misma posición. Pues se recuerda que el cilindro estará en la misma posición cada $\delta$ imágenes, ya que, éste es el valor de la frecuencia de su movimiento. Si a cada posición del cilindro se le llama fase, entonces se puede decir que existirán $\delta$ fases (siendo $\delta$ un número entero), es decir, $\delta$ posiciones del cilindro distintas. Por ejemplo, si $\delta$ fuera 3, el instante en el que el cilindro esté arriba del todo, será la fase 1, cuando baje un poco, será la fase 2 y cuando esté abajo del todo será la fase 3. A continuación, se muestra una imagen para ilustrar el ejemplo.

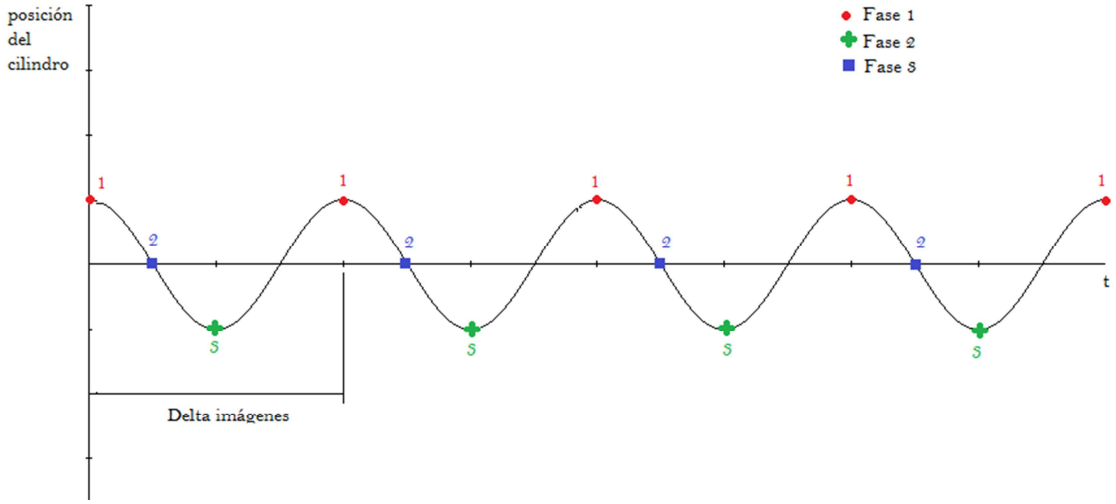



**Imagen 49. Esquema de las posiciones y fases del cilindro.**

Si se hace la media de todas las imágenes de una misma fase se obtendrá una nueva imagen con la posición de las partículas de esa fase. Si se repite esto con cada fase se tendrán $\delta$ imágenes con las partículas en cada posición. Si todas estas se vuelven a sumar se obtiene, de nuevo, el camino recorrido por las partículas en un ciclo. Siguiendo con el ejemplo anterior, el programa sumaria todas las imágenes de la fase 1 (punto) y haría la media, haría lo mismo para la fase 2 (cuadrado) y también lo haría con la fase 3 (estrella). Entonces se tendría 3 imágenes promediadas. Si se suman estás tres imágenes y se hace la media, se obtendría el recorrido que hicieron las partículas en las tres posiciones distintas.

Al final se obtendrán imágenes, donde se vean, más blancas, las sendas que recorrieron las partículas.

### 3.3.3. PIV (*Particle Image Velocimetry)*

En este último post-procesado, se va a realizar un PIV (*Particle Image Velocimetry)*. Se trata de una técnica de análisis que permite la obtención de un campo de vectores de velocidad. Para ello se ha modificado un software ya existente. Se ha utilizado un software gratuito, llamado LabPIV. Este software posee una interfaz gráfica, pero debido a la necesidad de realizar algunas modificaciones se ha extraído el código a Matlab. Ver anexo C.

El método PIV divide las imágenes en un conjunto de ventanas o recuadros. En un cuadro se escoge una partícula y se compara con la imagen siguiente. El programa calcula las probabilidades de que la partícula se haya podido desplazar a su alrededor y escoge la opción con mayor probabilidad. Para entenderlo mejor se puede ver la imagen 50. Repite esta operación con cada partícula de cada cuadro. Así, se obtiene la dirección y sentido de movimiento y permite la obtención de vectores.

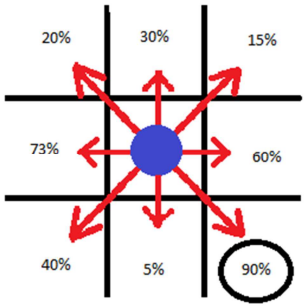



**Imagen 50. Esquema del cálculo de probabilidades en PIV.**

Nótese, que el método PIV funciona mejor cuantas más partículas haya disueltas en el líquido, pero se debe tener en cuenta que si se introducen demasiadas partículas en el agua se podrían producir variaciones en la reflexión de la luz al taparse unas con las otras.

Se han realizado modificaciones para que realice el PIV sobre cada fase y haga la media. De esto modo se reduce el error. Además, repite esta operación con dos experimentos separados pero que poseen la misma frecuencia y de nuevo hace la media. Así, el campo de vectores apenas tendrá errores. También, se exportan las variables de posición (xx, yy) y los vectores de velocidad media (uAVG, vAVG) que después se utilizarán para obtener resultados. Este programa se ha llamado '*PIV_promediado.m'*.

El campo de velocidades que se obtiene tiene unidades de pixel /frame.

### 3.3.4. Isolíneas de velocidades

Por último, se han utilizado los valores exportados del programa PIV. Las variables utilizadas son xx (posición en el eje x), yy (posición en el eje y), uAVG (velocidades en el eje x) y vAVG (velocidades en el eje y).

En el programa anterior, '*PIV_promediado.m*' no utiliza valores cualitativos con los que se pueda trabajar. Por ejemplo, los vectores de posición, xx e yy vienen dados en unidades de pixel. Así como el campo de vectores de velocidad tienen unidades de pixel por frame.

Para la transformación de unidades se ha calculado cuantos pixeles tiene un cilindro de diámetro. Para ello, en Matlab se pide que represente la imagen y se seleccionan dos puntos opuestos se pide que haga la resta de los dos vectores y se obtiene el número de pixels que existe entre esos dos puntos. Una vez conocido el diámetro en pixels se puede comparar con el diámetro real del cilindro y sacar el factor de conversión, que será:

[#] $1\ cm = 93.5\ pixels$

Por otro lado, se sabe que la cámara realiza 30 frames por segundo, así, la velocidad es:

[##] $1\frac{cm}{s} = \frac{93.5}{30}\frac{pixel}{frame}$

El programa *'contour_velocity.m'* realiza un cambio de unidades transformando los valores de posición en metros y la velocidad en metros/segundo. Además, representa iso-líneas de la velocidad respecto de los ejes de posición x e y. Representará iso-líneas el valor absoluto de la velocidad en la dirección x (uAVG), el valor absoluto de la velocidad en dirección y (vAVG) así como el valor absoluto de la velocidad total (v) donde v es:

[###] $$v = \left|\sqrt{\mathrm{uAVG}^2 + \mathrm{vAVG}^2}\right|$$

Este programa se ha adjuntado en el anexo D.

## 3.4. Resultados experimentales

A modo de resumen se presentan en la siguiente tabla los parámetros de los experimentos analizados.

**Tabla 3. Valores de los parámetros de los experimentos.**

| Experimento | Nº Imágenes | Valor $\delta$ | Valor frecuencia [Hz] | RS | $\epsilon$ | ga |
|---|---|---|---|---|---|---|
| a | 568 | 10.8932 | 2.754 | 70 | 0.2 | 2·0.01 |
| b | 632 | 10.8932 | 2.754 | 70 | 0.2 | 2·0.01 |
| c | 609 | 8.3963 | 3.57 | 90 | 0.2 | 2·0.01 |
| d | 608 | 8.19 | 3.663 | 90 | 0.2 | 2·0.01 |

No se han utilizado valores de Rs mayores, pues existe riesgo de que el fluido se comporte de forma turbulenta. Por otro lado, los valores de $\epsilon$ que se podrían haber utilizado podrían haber sido mayores. Sin embargo, para los mismos valores de Rs, cuanto menor son los valores de $\epsilon$ mayores pueden ser los valores de la frecuencia. Esto resulta interesante, porque recordando lo que se dijo al inicio del capítulo 3, la fuente de alimentación puede trabajar desde 0 a 30 Voltios. Si se quiere utilizar una frecuencia de 1,5 Hz, la fuente trabaja a 2 Voltios aproximadamente. Este valor es muy pequeño y hace que la fuente se vuelva inestable y no de una frecuencia constante. Por eso, se han utilizado valores de $\epsilon$ menores para poder utilizar una frecuencia de trabajo mayor.

### 3.4.1. Sendas de las partículas

Tal y como se comentó anteriormente existirán 10 posiciones diferentes del cilindro para los experimentos a y b, y 8 posiciones para los experimentos c y d. A continuación, a modo de ejemplo, sólo se muestra una imagen de la misma fase.

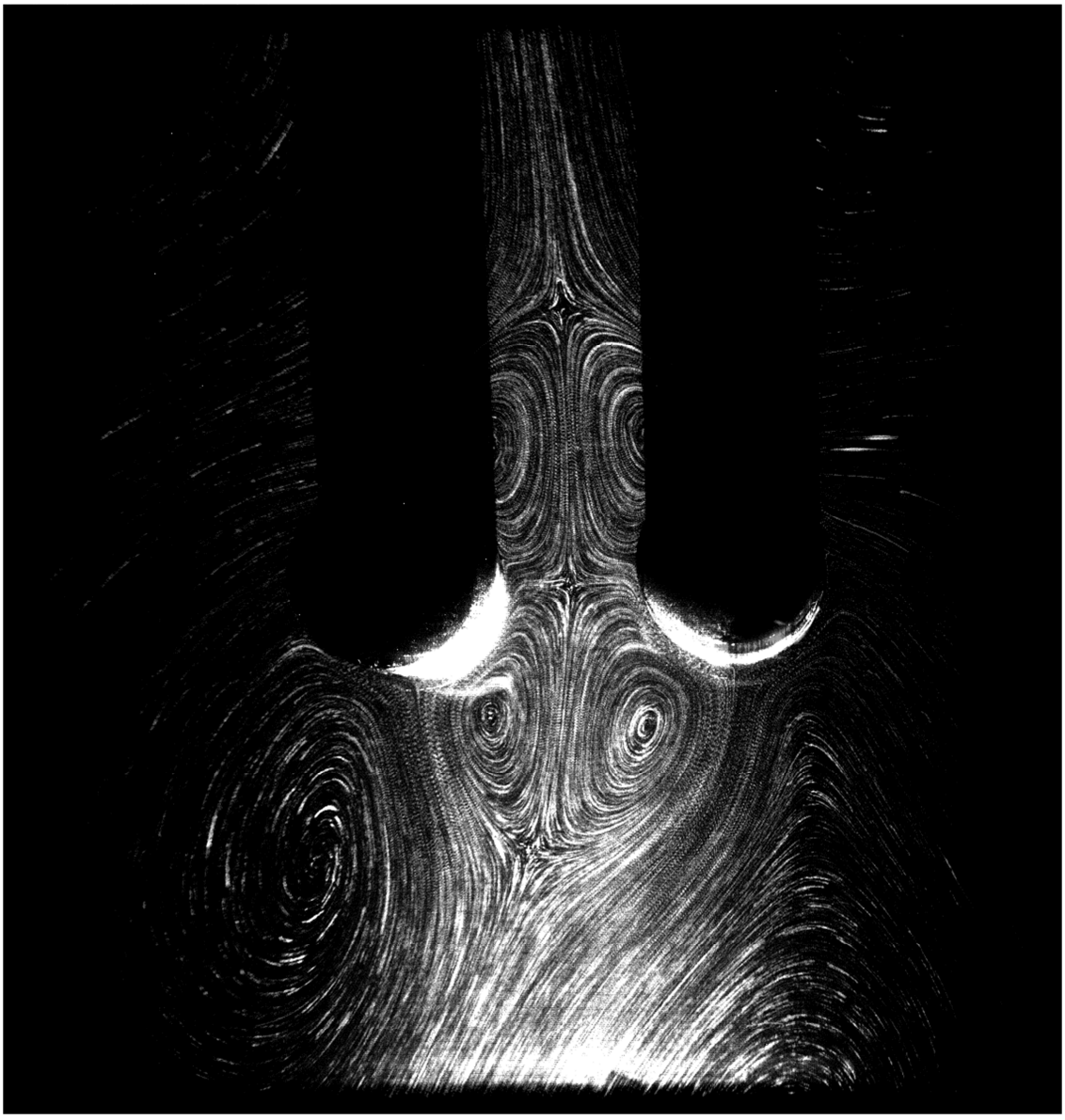


**Imagen 51. Sendas experimento a.**

**Imagen 52. Sendas experimento b.**

**Imagen 53. Sendas experimento c.**

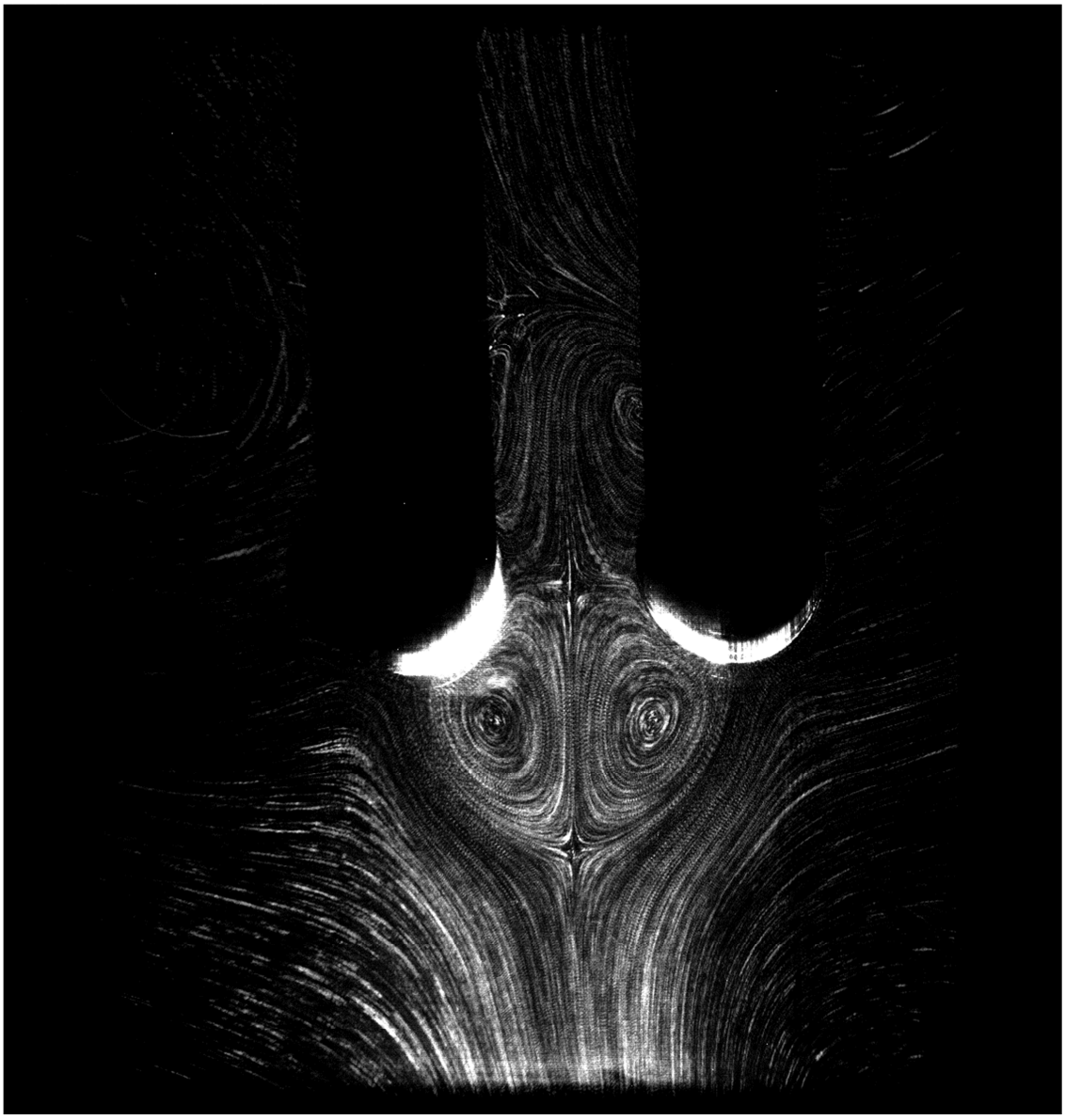


**Imagen 54. Sendas experimento d.**

Se puede ver que el experimento **a** no es completamente simétrico. Pero, se ve claramente como en los cuatro experimentos aparecen 4 celdas de recirculaciones en la zona central, y que son simétricos respecto del eje x como del y.

Además, se observa como las partículas son atraídas hacia las paredes del cilindro y recorren su superficie hasta ser expulsadas. Cuando son expulsadas lo hacen a través de un chorro cuya dirección es claramente hacía la del otro cilindro y son empujadas por los vórtices.

El recorrido de la partícula se muestra en la siguiente imagen.

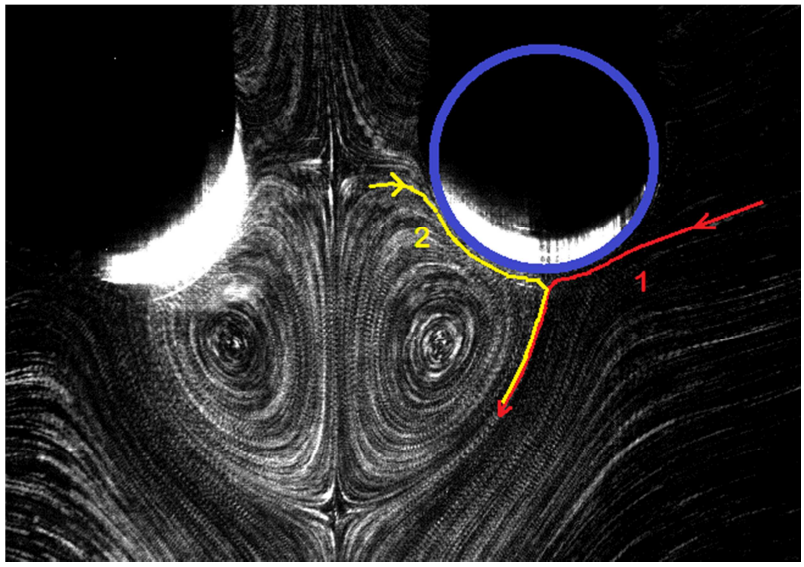



**Imagen 55. Recorrido de una partícula en el tiempo.**

Se muestra que el chorro va en dirección al eje de simetría de los cilindros y se forman recirculaciones. Esto hecho es la primera evidencia de que la hipótesis realizada por **Riley y Coenen** (2009) que predecía que los chorros tendrían sentidos opuestos es falsa.

### 3.4.2. PIV (*Particle Image Velocimetry)*

A continuación se muestran los campos de velocidades de los experimentos a y b.

Se ha de tener en cuenta que las imágenes salen giradas 180º respecto del eje horizontal. Además, se ha generado vectores de velocidad encima de los cilindros. Lógicamente, estos vectores no son reales pues en esas zonas no existe movimiento de partículas de vidrio. Surgen porque Matlab ha detectado diferencias en los niveles de color en esas zonas y los procesa como si existieran partículas aunque no haya.

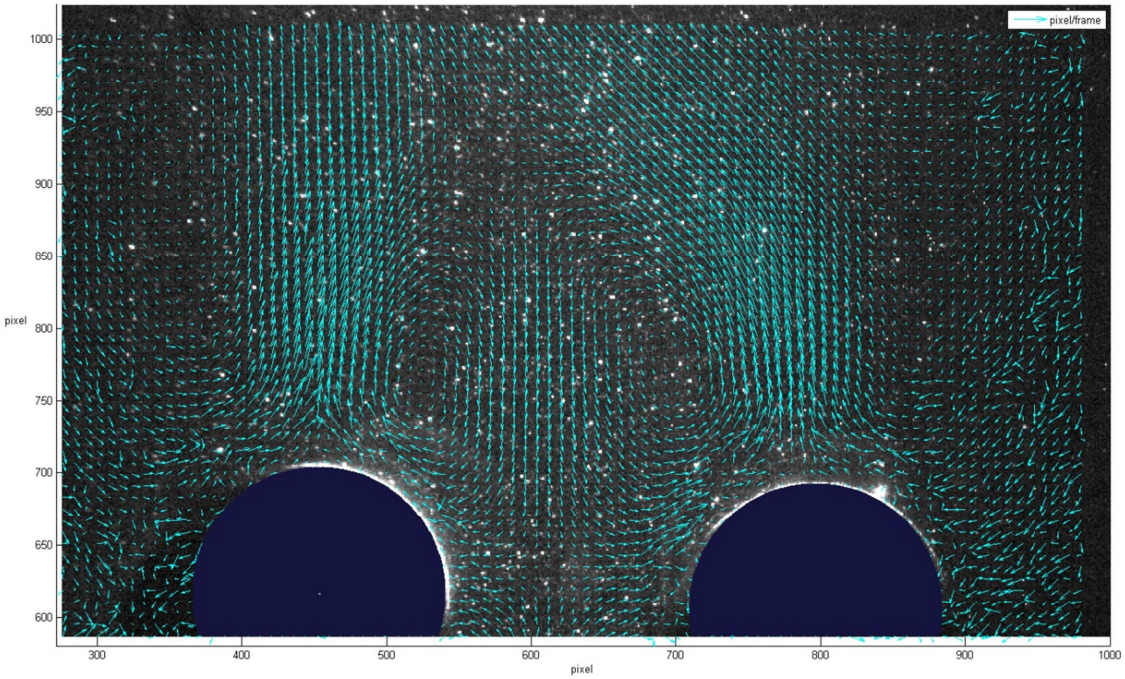



**Imagen 56. PIV experimento a.**

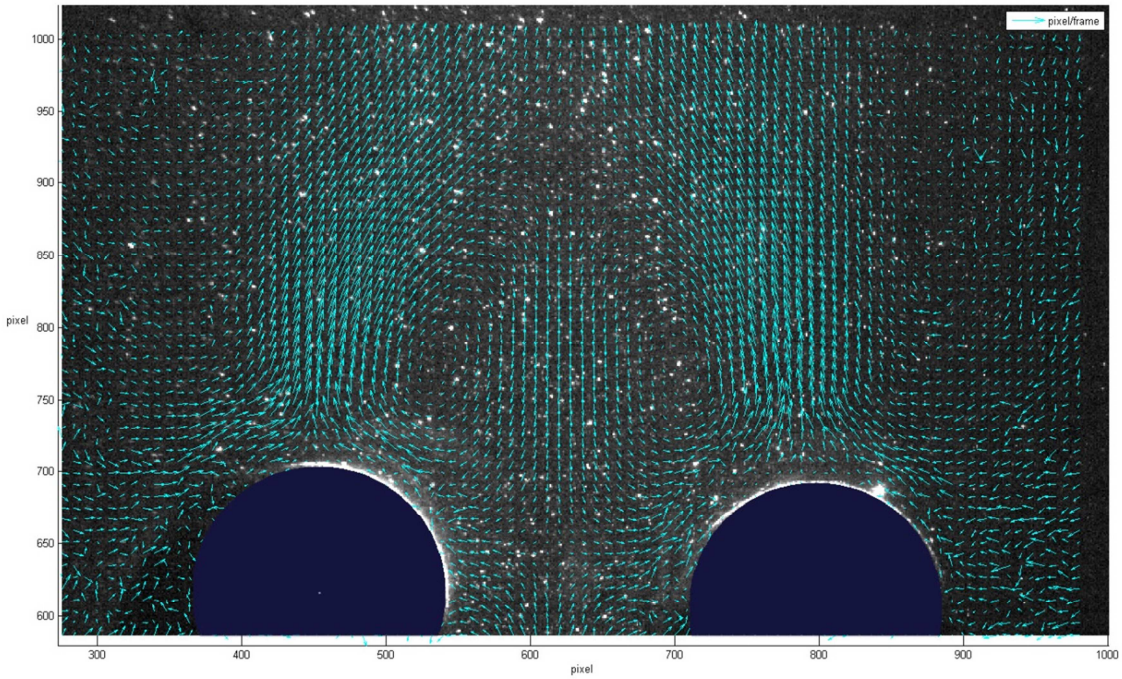



**Imagen 57. PIV experimento b.**

También, se ha realizado el PIV haciendo la media de los dos experimentos anteriores.

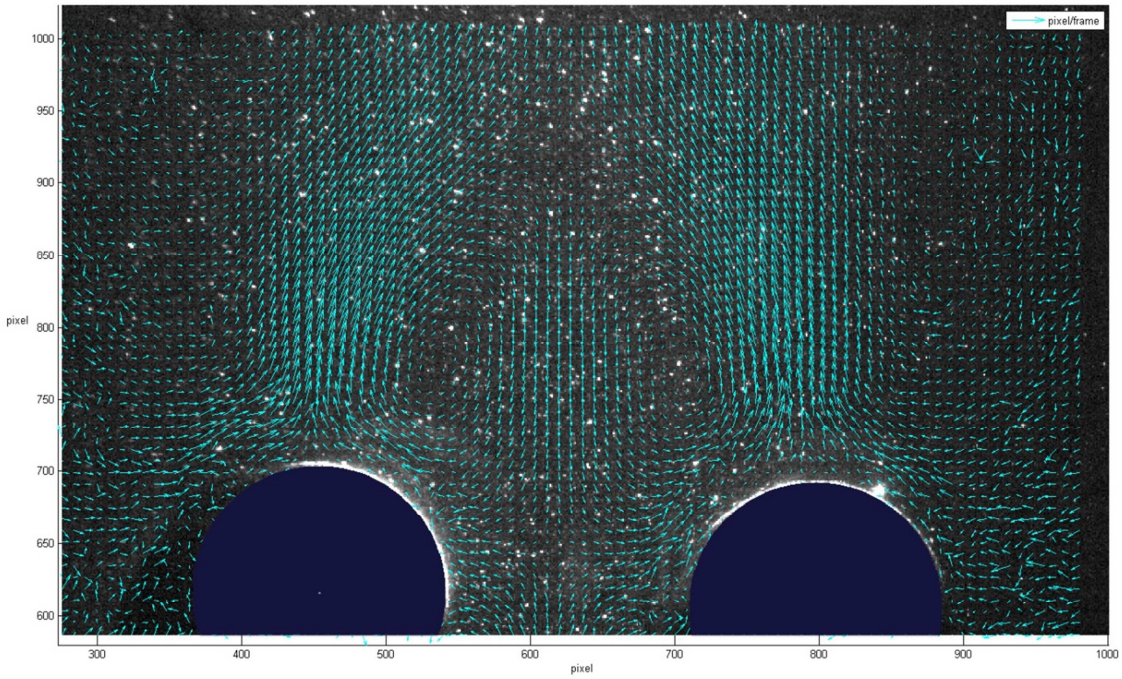



**Imagen 58. PIV experimentos a y b promediados.**

Se puede ver que apenas hay diferencias.

Lo mismo se hará con los casos c y d.

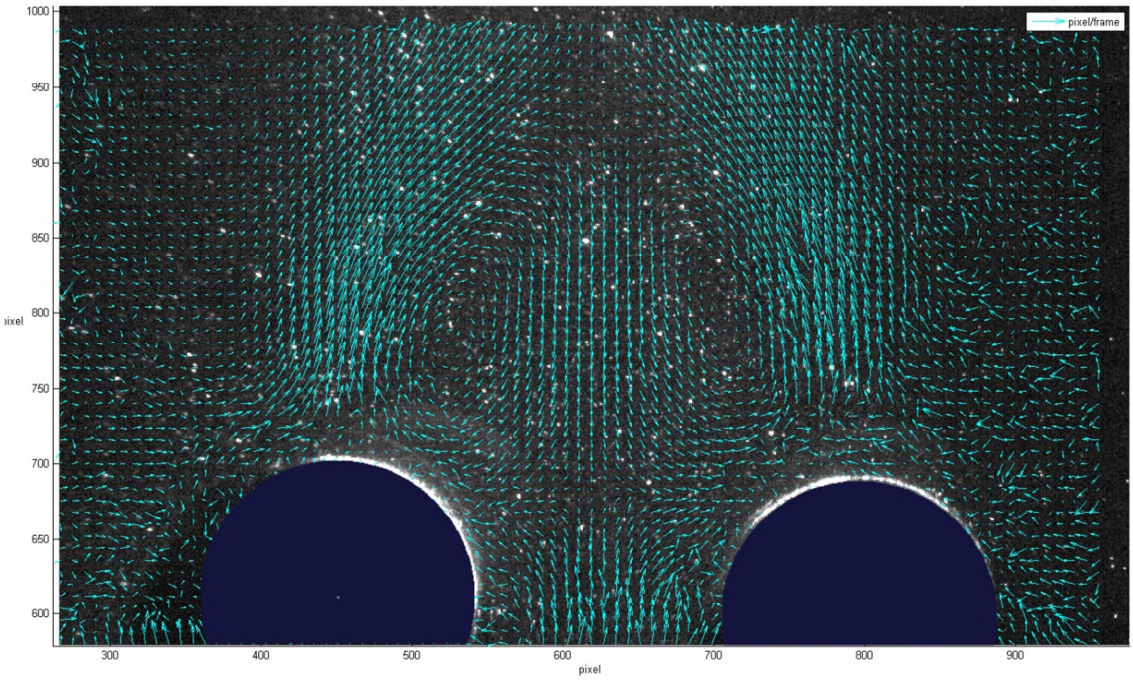



**Imagen 59. PIV experimento c.**

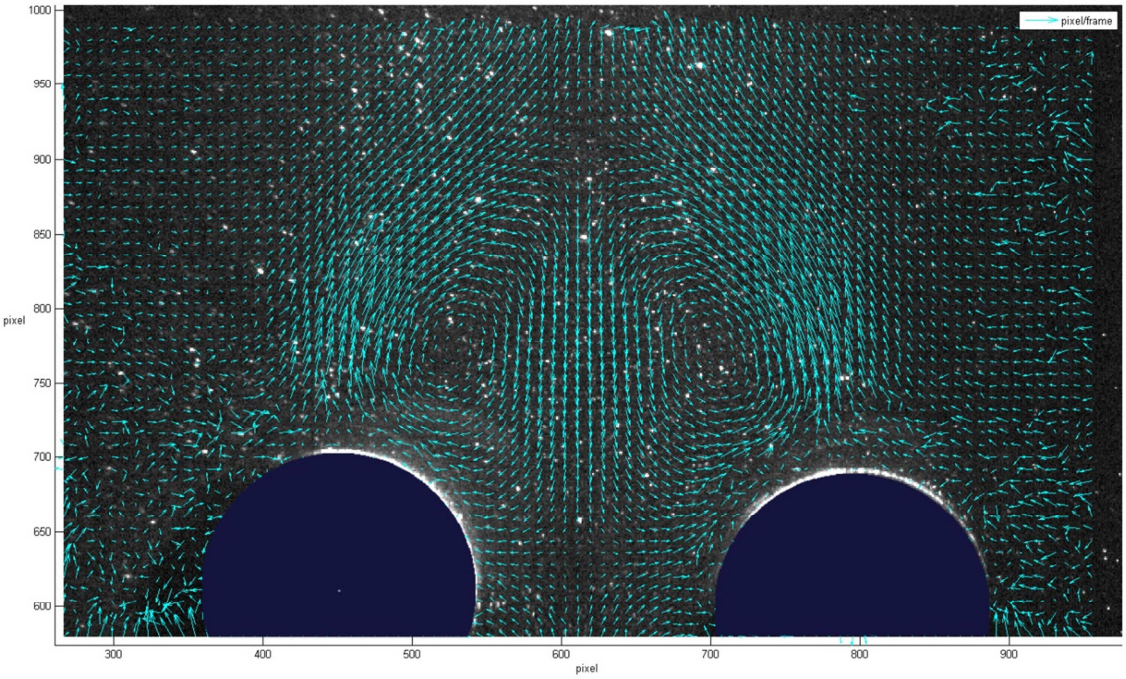



**Imagen 60.PIV experimento d.**

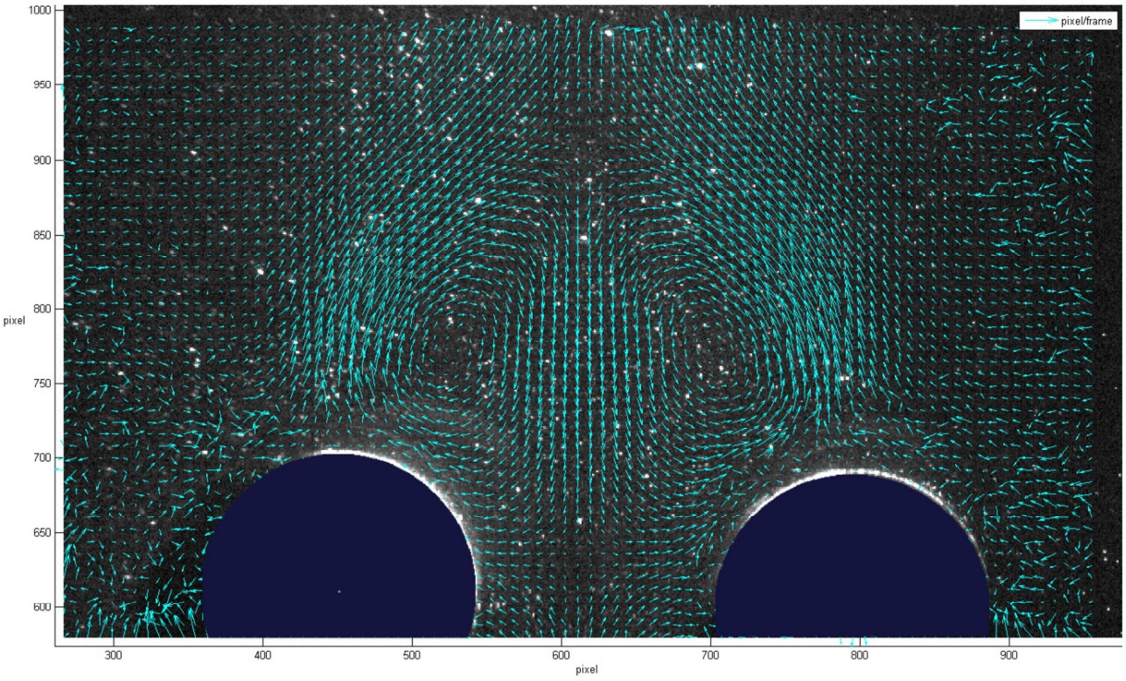



**Imagen 61. PIV experimentos c y d promediado.**

Al igual que en el caso anterior apenas hay diferencias entre los PIV de cada experimento respecto del promedio.

Estos campos de velocidades añaden información a las sendas obtenidas en el post-procesado anterior. Por un lado, se observa, que como en los análisis anteriores, se producen zonas de recirculación. Pero además, se sabe que en las zonas donde los vectores son muy pequeños o

inexistentes es debido a que no se produce movimiento. Estas partículas sin movimiento, se encuentran en las zonas alejadas de los cilindros (a excepción de las partículas que son empujadas por el chorro) y las partículas que se están en el centro de las zonas de recirculación.

Por tanto, no sólo se puede afirmar que existen recirculaciones, sino que las partículas que están en el centro de éstas, permanecen estáticas. Es decir, se produce un atrapamiento de partículas.

Se puede afirmar, por tanto, que la hipótesis de **Riley y Coenen** (2009), sobre la dirección de los chorros no se cumple experimentalmente.

### 3.4.3. Isolíneas de velocidades

Se muestran a continuación las iso-líneas de velocidad del caso b. El resto de casos son similares, por ello no se han incluido las imágenes.

Se ha pintado los cilindros como superficies macizas para tener una referencia.

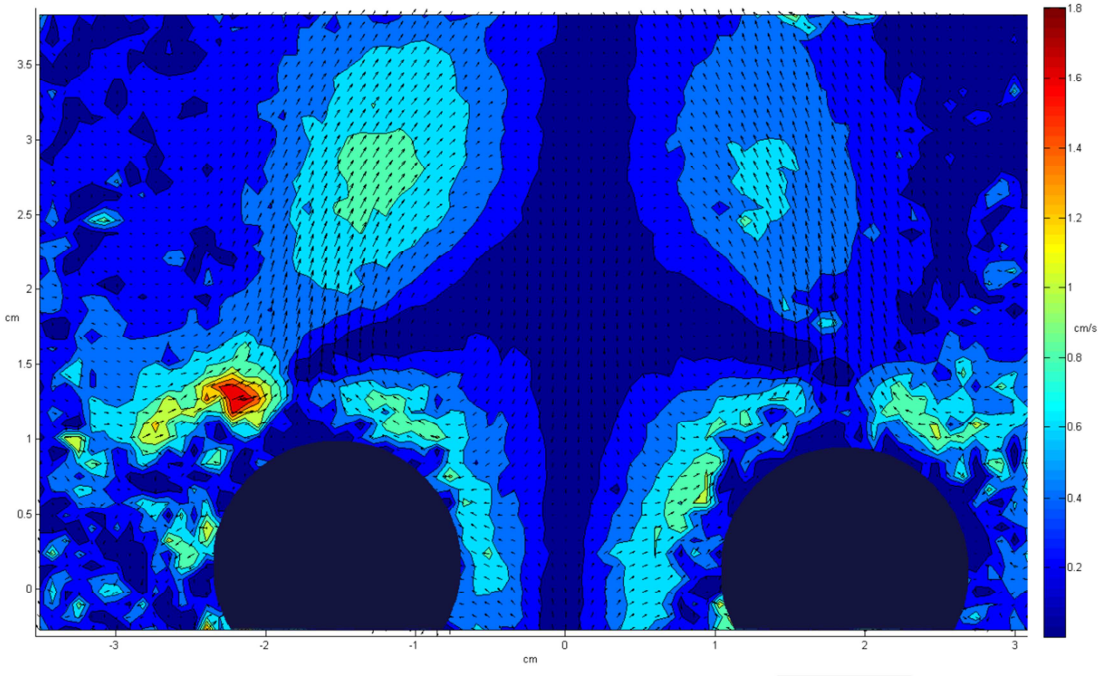



**Imagen 62. Iso-líneas de la velocidad en la dirección x.**

En la imagen 62, se ha representado iso-líneas de la magnitud velocidad en dirección x respecto de los ejes de posición x e y. Donde la posición tiene unidades de cm y la velocidad de cm/s.

En esta gráfica se puede ver como el fluido que está más alejado del eje de simetría tiene mayor velocidad. Esto provoca, que el chorro vaya dirigido hacia el interior de los dos cilindros.

Sólo se ha calculado la parte superior de los dos cilindros, pero como ya se ha comentado antes, existe simetría no sólo respecto del eje y sino del x.

Además, esta imagen permite ver el punto de eyección. Este punto, se muestra en detalle en la siguiente imagen.

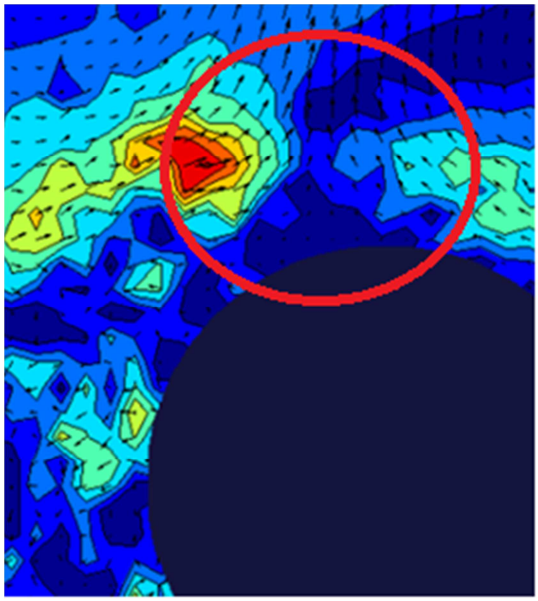


**Imagen 63. Punto de eyección.**

La diferencia de color indica que hay un cambio en las velocidades en el punto donde se cruzan es el punto de eyección.

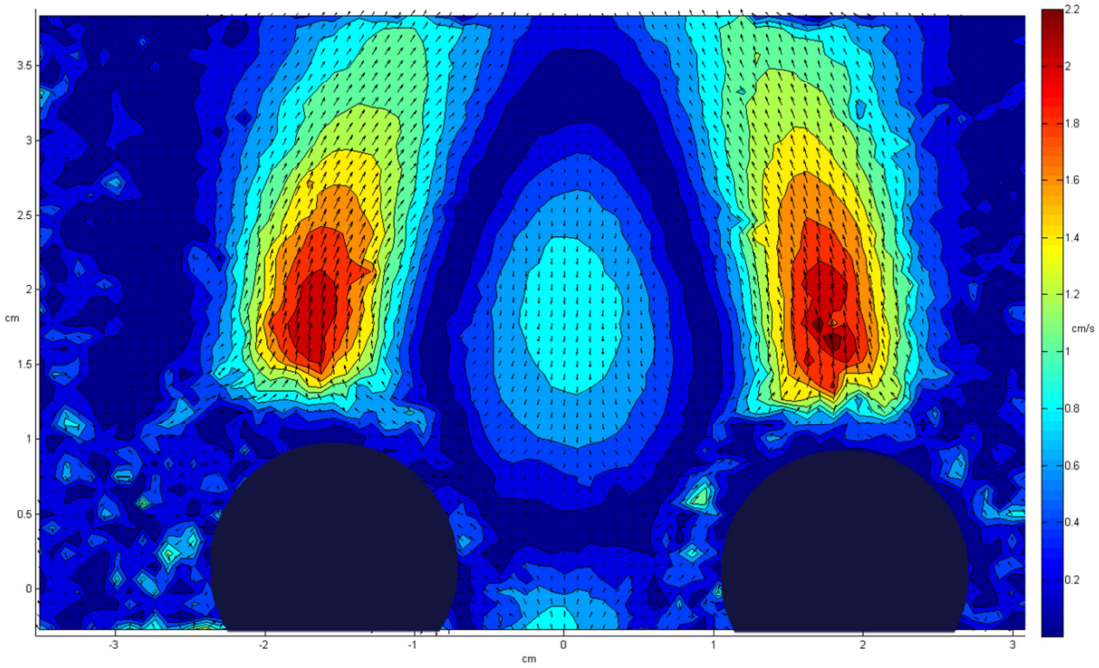



**Imagen 64. Iso-líneas de la magnitud dela velocidad en dirección y.**

En la imagen 64, se muestra las iso-líneas de la magnitud de la componente de la velocidad en dirección y respecto de los ejes x e y.

Se puede observar, que las zonas de mayor velocidad son aquellas cercanas a las superficies de los cilindros donde se inicia la formación y eyección del chorro.

Por otro lado, se puede observar, que tal y como se comprobó mediante el método PIV, en el interior de las recirculaciones la velocidad es casi nula y no se produce movimiento.

Por último, resulta muy interesante como el chorro no se forma justo en la propia superficie del cilindro, sino un poco más alejado de ésta. Esto demuestra la existencia de una capa de Stokes, muy fina y una capa Límite donde se produce el efecto *steady streaming* un poco más alejada de la superficie. No se puede cuantificar el espesor de esta capa de manera experimental por ser muy fina. Pero si se demuestra su existencia. A continuación se muestra un detalle de lo anteriormente descrito.

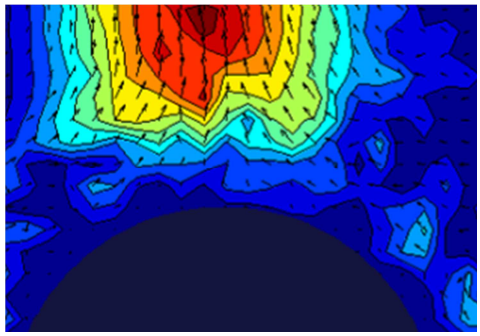


**Imagen 65. Detalle de las capas de Stokes y Límite.**

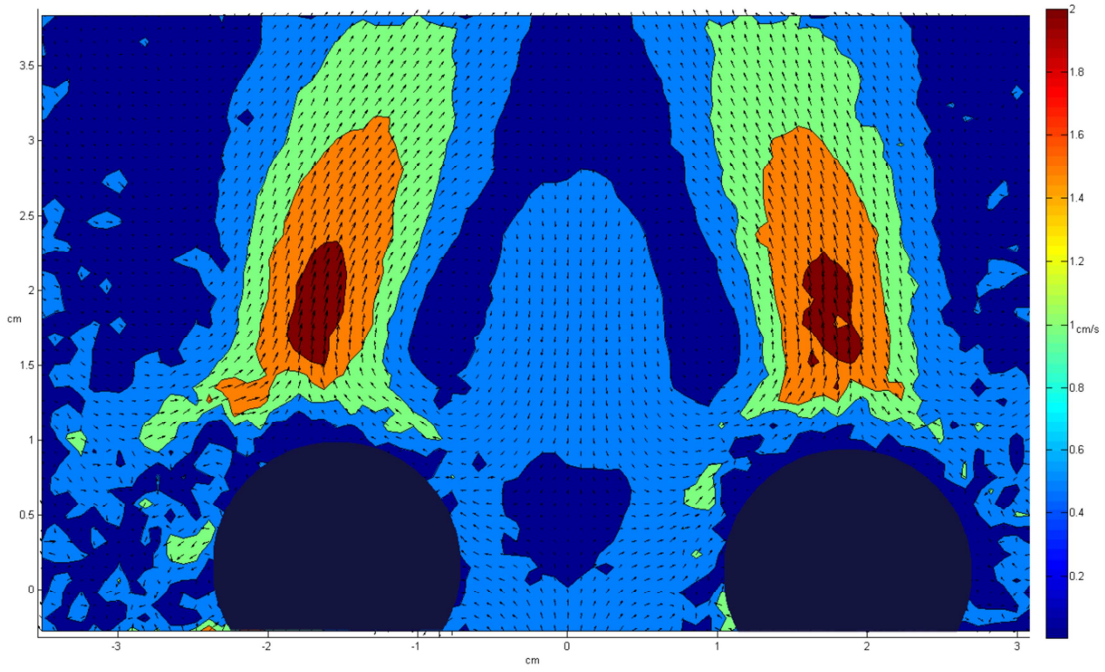



**Imagen 66. Iso-líneas de la magnitud de la velocidad absoluta.**

Por último, se han representado las iso-líneas de la velocidad respecto de los ejes x e y.

Esta imagen corrobora lo explicado anteriormente. Las zonas de mayor velocidad son aquellas en las que el chorro es expulsado hacia el exterior de la superficie del cilindro. En el interior de las recirculación no hay velocidad ninguna. Y además, el chorro se forma en la capa Límite, un poco alejado de la superficie del cilindro.

### 3.4.4. Comparación entre sendas y PIV

Se ha querido superponer una imagen de las sendas con el campo de vectores de velocidad PIV para demostrar que las líneas coinciden a la perfección. Siguiendo con el caso usado en el apartado anterior, se ha utilizado ambas imágenes del experimento b.

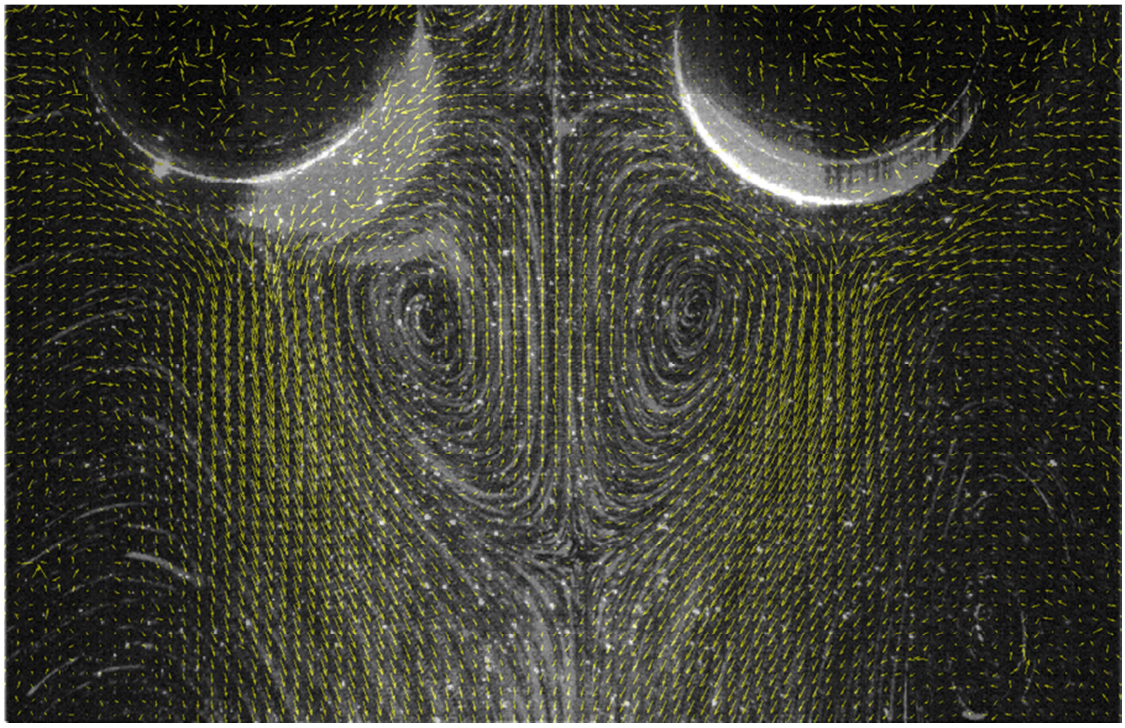


**Imagen 67. Superposición de sendas y PIV del experimento b.**

Se puede ver, que las líneas de vectores coinciden con las sendas de las partículas de vidrio. Y se ven claramente las celdas de recirculación. Además, se comprueba que en las zonas de recirculación, coincide que no hay vectores o son muy pequeños, lo que significa que no existe movimiento en esas celdas.

# 4. DESARROLLO COMPUTACIONAL

# 4. DESARROLLO COMPUTACIONAL

Toda simulación hecha por ordenador necesita ecuaciones matemáticas que modelen el problema que se desea estudiar. En este caso, se van a utilizar las ecuaciones de Navier-Stokes de cantidad de movimiento y continuidad.

Si las simulaciones coinciden con los resultados obtenidos experimentalmente, se darán por demostradas estas teorías.

Para los ensayos computacionales se ha utilizado un programa de simulación de fluidos o *CFD (Computacional Fluid Dynamics)* llamado Fluent, que se espera, den resultados similares a los obtenidos en el laboratorio. Además, como alternativa a Fluent, se presentan los resultados obtenidos por otro programa de *CFD* llamado Gerris.

Tanto Fluent como Gerris, necesitan mucho tiempo para procesar los casos y obtener soluciones. Este tiempo puede variar desde horas, si los casos son sencillos, hasta semanas. Se necesitaría tener el ordenador encendido durante días sin poder usarlo para otra cosa. Por ello, se ha utilizad un clúster. Se define clúster como un conjunto de computadores que utilizan un hardware común y que se gestionan como si fuera una sólo computador, gracias a un gestor de clúster. Para enviar la información al clúster es necesario tener la información en la red del clúster. Además, es necesario escribir un archivo cabecera, donde se indicará que archivo se debe procesar y en qué ficheros se exportarán los datos. Estos datos exportados son devueltos por el procesador para guardarlos en la red. De ahí, el usuario deberá descargarlo a su ordenador personal. El acceso al clúster está limitado a los usuarios que estén registrados. A continuación, se adjunta un diagrama del funcionamiento del clúster.

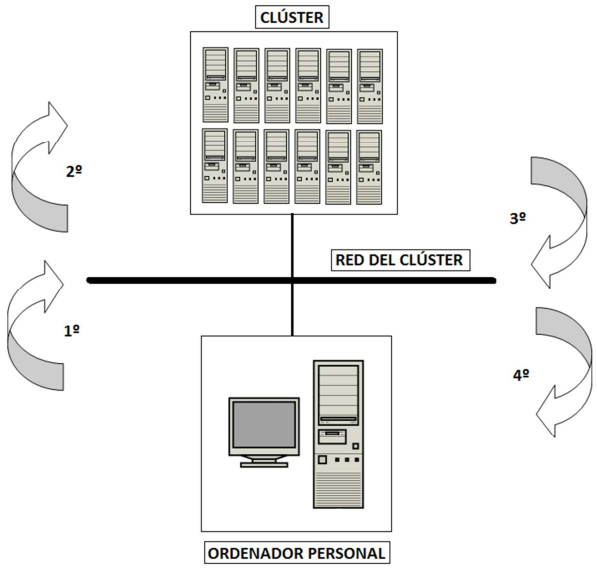



**Imagen 68. Diagrama de trabajo con el clúster.**

Los casos que se han simulado con ambos programas son, lógicamente, los mismos que los realizados experimentalmente, para poder comprar los resultados. Sin embargo, a diferencia

de en el análisis experimental donde se movían los cilindros, durante las simulaciones computacionales lo que se mueve es el fluido. Se recuerda, que ambos casos son equivalentes pero, se ha escogido simular el movimiento del fluido mientras los cilindros permanecen quietos por simplicidad, ya que, de otro modo, se debería haber creado una malla dinámica que se moviera a la vez que lo hicieran los cilindros. Es una opción mucho más compleja que la de otorgar una velocidad variable al fluido.

Los casos simulados con Fluent son los siguientes:

**Tabla 4. Parámetro de las simulaciones.**

| RS | $\epsilon$ | a | ga |
|---|---|---|---|
| 70 | 0.2 | 0.01 | 2·0.01 |
| 90 | 0.2 | 0.01 | 2·0.01 |

## 4.1. Fluent

Fluent es un programa que pertenece al paquete Ansys. Se trata de un software que permite modelar flujos, turbulencias, transferencias de calor o reacciones para aplicaciones industriales. Posee una interfaz gráfica, lo cual, hace más fácil su uso y análisis de resultados. Además, incluye módulos como por ejemplo, el de combustión que permite simular el comportamiento de un motor de forma sencilla. Sin embargo, para el procesado de variables dependientes del tiempo es necesario incluir UDFs *(UserDefinedFunctions),* que son líneas de código en lenguaje C que se deben incorporar a Fluent. Más adelante, se explicará cual es el código usado y cómo acoplarlo a Fluent.

Para generar la simulación es necesario seguir los siguientes pasos:

- Crear la geometría
- Crear la malla
- Ajustar parámetros en Fluent

Existe un programa que permite agrupar todos los programas necesarios para llevar a cabo estos pasos.

Este programa se llama Workbench. Se trata de un software, perteneciente al paquete de programas Ansys. Este programa es un entorno que mediante una interfaz gráfica permite agrupar y llamar a otros programas.

Tal y como se ve en la imagen 69, en la columna de la izquierda se puede escoger 'paquetes' específicos según lo que se desee hacer. En este caso, se ha escogido el paquete '*Fluid Flow (Fluent)'.* Este paquete incluye un programa para la generación de la geometría, otro para crear la malla y por último permite modificar los parámetros de la simulación mediante Fluent.

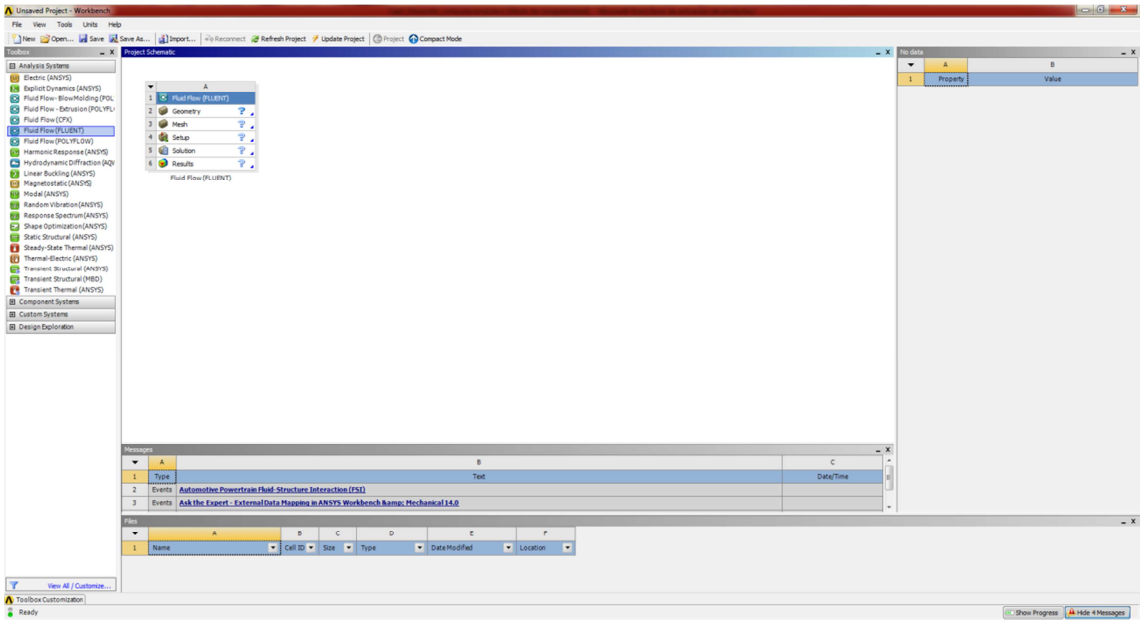


**Imagen 69. Entorno de Workbench.**

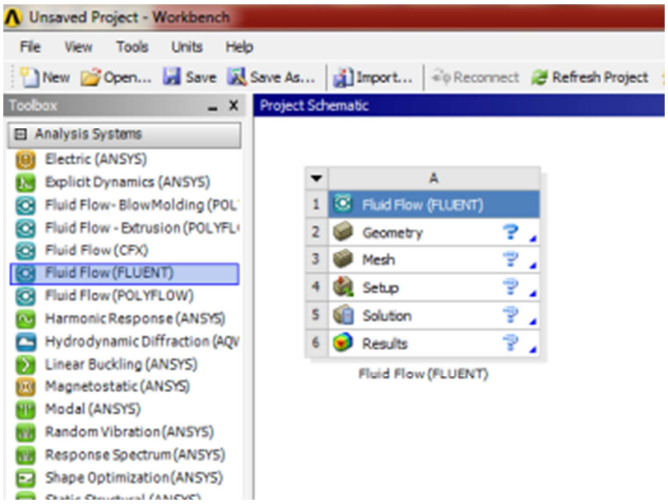



**Imagen 70. Opciones del módulo Fluid Flow de Worckbench.**

### 4.1.1. Crear la geometría

El programa Workbench llama al programa *DesignModeler* cuando se pulsa en el botón *Geometry*. Es un programa sencillo que permite dibujar tanto en dos como en tres dimensiones. La geometría que se cree será la que luego analice el programa Fluent.

Sólo se simulará un cilindro, por tratarse de un caso simétrico. De este modo se ahorran recursos y tiempo de procesado.

Como lo que se desea estudiar es el comportamiento del fluido, se dibuja la superficie que rodea al cilindro, por donde pasará el fluido. Así pues, el cilindro, en realidad, es una parte hueca en el dibujo.

La superficie que se dibuje debe ser muy grande comparado con el radio del cilindro (10 veces mayor), aunque la zona que se desee analizar sea más pequeña. Esto es así, para que la zona de interés sea analizada por completo y no se vea comprometida por ser una condición de contorno (un extremo en la geometría).

Se ha optado por la siguiente geometría. Ver imagen 71.

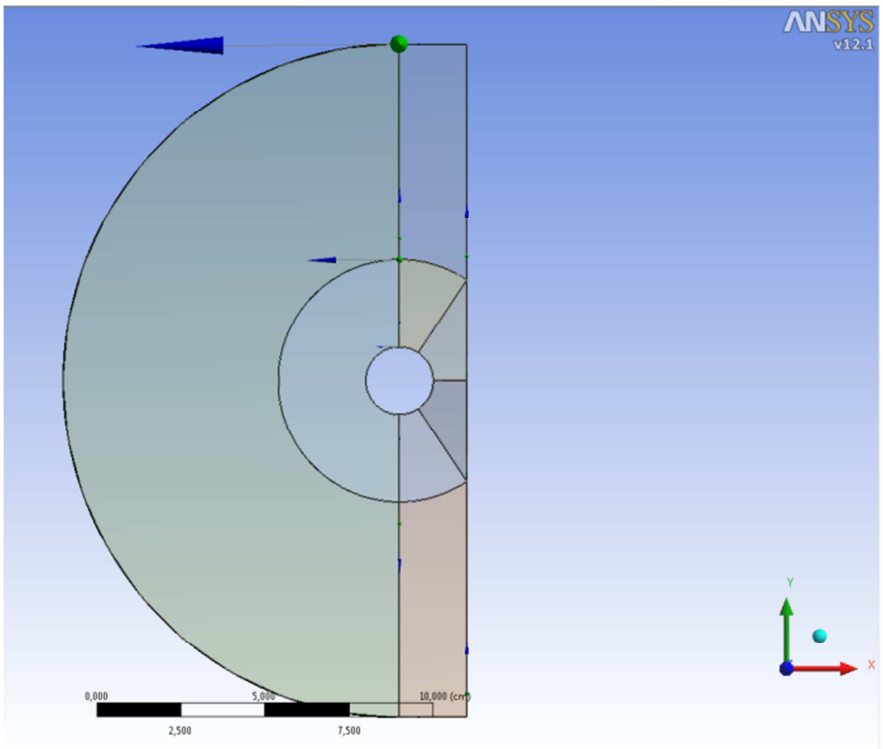



**Imagen 71. Geometría.**

Como se ve, se ha escogido una geometría semicircular dividida en 8 partes. Las partes se realizan para ayudar a crear la malla en el siguiente paso. Ha de aclararse que aunque la geometría esté dividida, se procesarán todas las partes como si fueran una sola.

En el siguiente esquema, se han dibujado las cotas de la geometría en función del radio **a**.

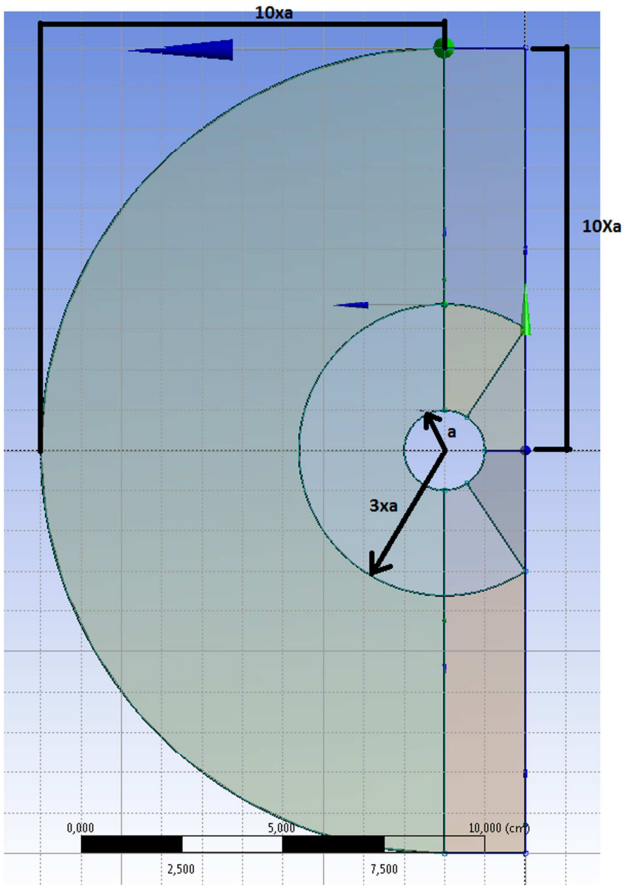



**Imagen 72. Geometría con cotas en función del radio a.**

### 4.1.2. Crear la malla

Se llama malla porque un conjunto de líneas se cruzan entre ellas. Los puntos donde intersectan las rectas se llaman nodos. En cada uno de estos nodos se aplicarán las ecuaciones que se escojan en el programa Fluent. Por tanto, cuantos más nodos más ecuaciones se realizan y por tanto mejor definida quedará la solución. Sin embargo, debe existir un equilibrio entre el número de nodos y el tiempo de procesamiento. Pues éste, aumentará con el aumento de número de nodos.

Cuando se habla de refinar una malla se refiere a aumentar el número de nodos. Si la malla está poco refinada el problema no quedará bien definido y la solución será errónea. Esto es así porque, si entre los nodos el fluido tiene un comportamiento de interés no podrá analizarse. Este hecho se ilustra en el siguiente dibujo.

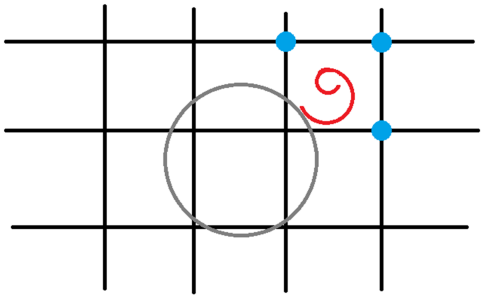


**Imagen 73. Ejemplo de malla errónea.**

Como el efecto a estudiar, en este caso es una recirculación, ocurre entre los nodos (puntos más grandes) no es captado y por tanto la malla no es adecuada. Así pues, para saber qué tipo de malla ha de emplearse es necesario un conocimiento previo del problema y una previsión de los resultados a obtener. Por ello, el análisis de estudios y artículos de otros autores se hace imprescindible.

Para dibujar la malla se pulsa el botón *Mesh,* el entorno Workbench abre automáticamente otro programa llamado *Meshing.*

La imagen 74, muestra la malla que se ha realizado.

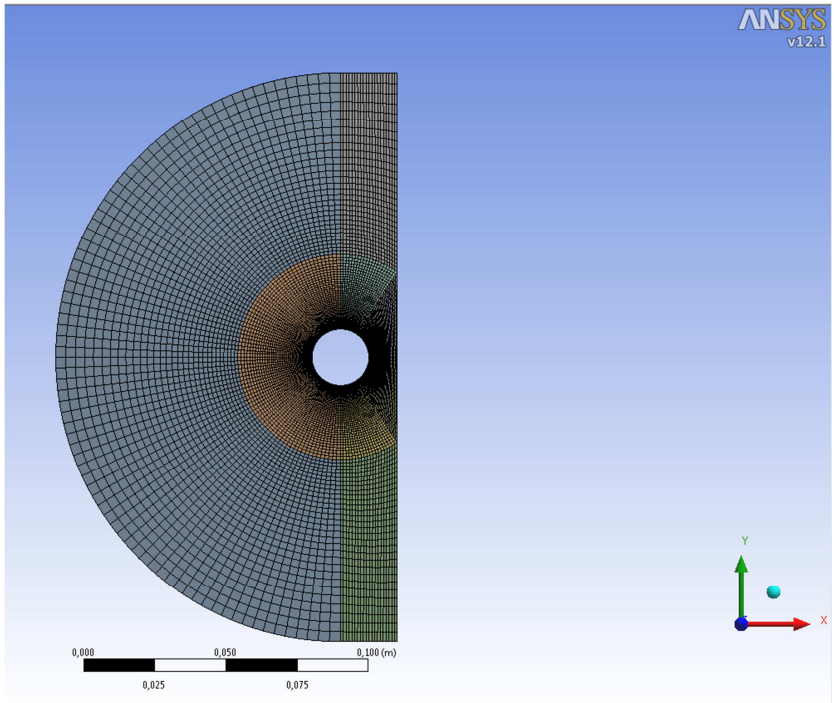



**Imagen 74. Malla**

Se ha optado por una malla con elementos cuadrados.

Se ha creado una malla estructurada. Se define malla estructurada a aquella en la que todos los elementos cumplen una norma. Para crear una malla estructurada y las celdas no se

deformen, cada lado tiene que tener el mismo número de elementos que su lado opuesto, de ahí, la necesidad de crear distintas divisiones en la geometría.

Se sabe que el efecto *Steady Streaming* y la formación de chorro se producen muy cerca de la superficie y que lo que ocurre muy lejos del cilindro carece de interés. Por ello, la malla es más refinada en las zonas cercanas a la pared y va aumentando de tamaño según se aleja hacia el exterior. Ver imagen 75.

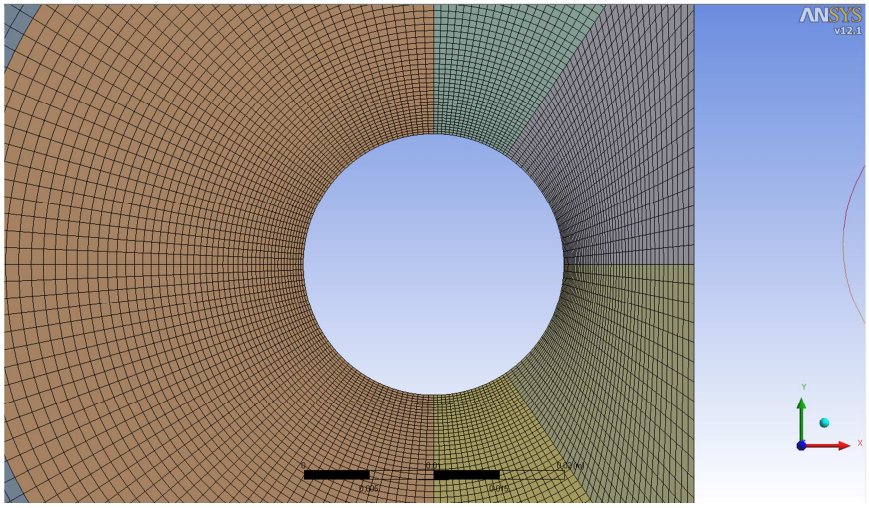



**Imagen 75. Malla cerca de la superficie del cilindro.**

El aumento de tamaño se consigue utilizando una opción llamada *bias* que permite introducir un factor de crecimiento o disminución de las celdas en el espacio. Esta es otra razón por la que se han creado distintas partes en la malla. No a todas las partes de la malla, se les ha dado el mismo factor *bias.* Por ejemplo, los elementos de la semicircunferencia interior, que se ve en la imagen 75, poseen un factor *bias* mayor, porque interesa que tengan menor tamaño cuanto más cerca estén de la pared del cilindro.

Además, para mayor precisión las celdas más cercanas al cilindro se refinarán en el siguiente paso. Fluent ofrece una opción para dividir celdas. Lo que hace es dividir una celda en cuatro más pequeñas. Es decir que de tener una celda se pasa a tener $2^2$. Si se repite esta operación, esa celda pasará a tener $2^4$ celdas, y así sucesivamente.

En este caso, se ha repetido esta división tres veces, de modo que al final, cada celda pegada a la pared se ha convertido en $2^6$ celdas.

Esta operación se hace pulsando en *Adapt > Boundary.*

A continuación se muestra una imagen de esa zona.

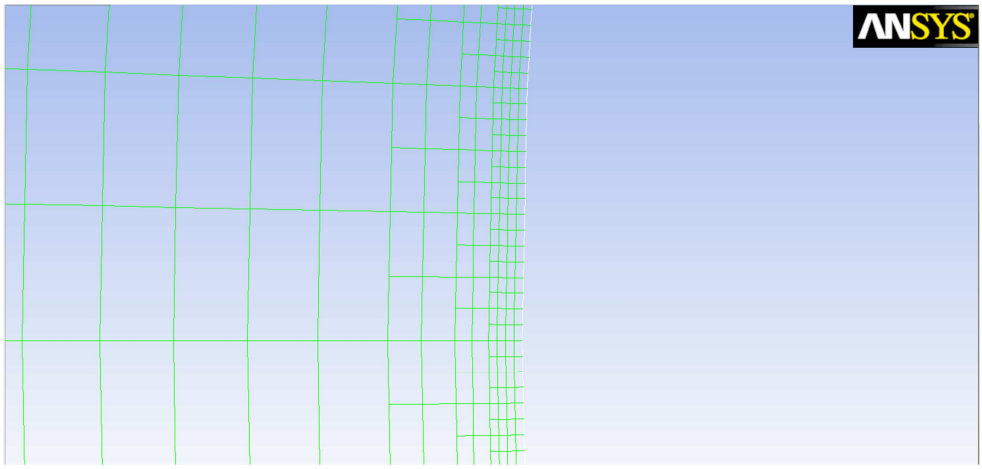



**Imagen 76. Refinado con Fluent.**

Además, se ha tenido especial cuidado en las uniones. Los elementos de la malla a cada lado de las divisiones deben tener el mismo tamaño para que no se produzcan alteraciones en las ecuaciones que pudieran llevar a un resultado erróneo. En la imagen 77, se ve una muestra de cómo son estas uniones.

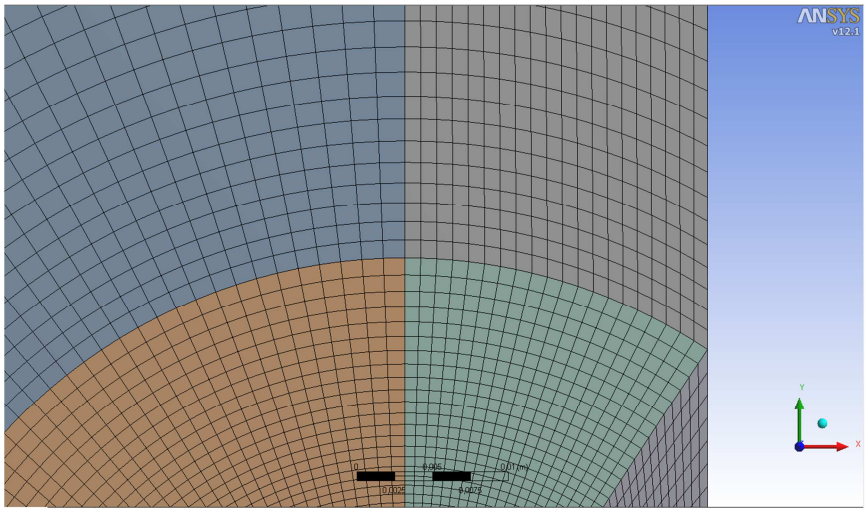



**Imagen 77. Uniones en la malla.**

Las dimensiones de la malla final se muestran a continuación.

**Tabla 5. Tamaño de la malla.**

| | |
|---|---|
| **Nodos** | 50.240 |
| **Elementos** | 52.860 |
| **Caras** | 103.100 |

Una vez creada la malla, se le asignarán unos nombres a cada contorno. Después, estos nombres aparecerán en Fluent y se le podrá asignar una condición de contorno distinta a cada parte.

En la imagen 78, se ve cómo se han asignado 4 nombres. A la zona de la circunferencia se la ha llamado *Wall* (pared), la zona exterior recibe el nombre de *Velocity_inlet* (Velocidad de entrada) y por último, el eje de simetría se ha denominado *Axis (*eje).

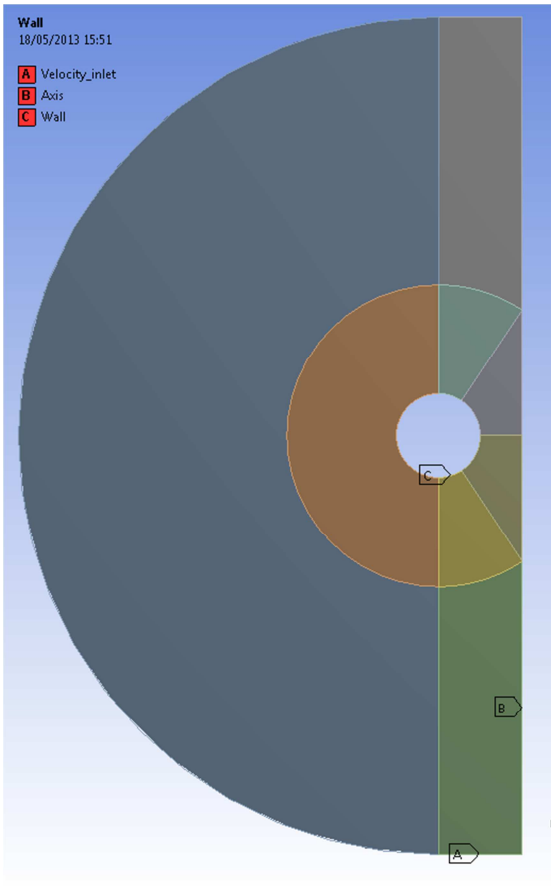



**Imagen 78. Nombre de las distintas partes de la malla.**

En ocasiones, puede crearse una malla que no sea adecuada del todo y producir unos resultados. Estos resultados, a simple vista, puede parecer que están bien. Pero han de analizarse en detalle. En caso de que los resultados sean erróneos, ha de volverse a crear otra malla.

Para saber si la malla que se ha usado se adecúa al problema, se ha resuelto en Fluent. Los resultados se han comparado con los resultados experimentales. En ambos casos coinciden. Sin embargo, la forma más adecuada consiste en crear una malla más refinada que la inicial y resolver la simulación. Después, se compararán ambos resultados. Si son diferentes, significa que la malla está influyendo en los resultados y por tanto no es adecuada. Los resultados deben ser siempre independientes de la malla.

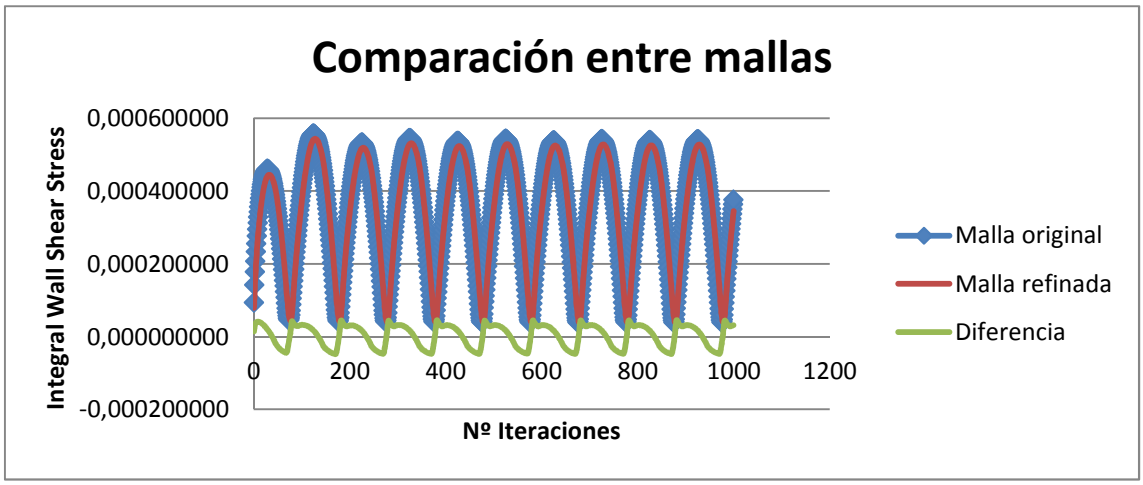
Para la comprobación se ha duplicado cada celda de la malla original. Además, se ha refinado la zona más cercana a la superficie, duplicando las celdas que se encuentran en un radio de 2cm del centro de la circunferencia. Se ha obtenido una malla del siguiente tamaño.

**Tabla 6. Tamaño malla refinada.**

| | |
|---|---|
| **Nodos** | 599.468 |
| **Elementos** | 609.784 |
| **Caras** | 1.209.252 |

Para la comparación entre la malla original y la refinada, se ha analizado la evolución de la variable de la fuerza tangencial del fluido sobre el cilindro frente al tiempo. No se ha escogido la fuerza longitudinal porque es la presión y, se sabe que, apenas variará con el tiempo.

Se añade a continuación, una gráfica en la que se muestra la evolución de la fuerza tangencial o en inglés (*Integral Wall Shear Stress)* a lo largo de 1000 iteraciones. Sólo se han mostrado 1000 iteraciones de las 24000 que se han realizado para simplificar, pero el resultado es común para el resto de iteraciones.




**Gráfica 1. Comparación entre malla original y refinada.**

Se observa que las gráficas de ambas mallas se superponen. La diferencia entre una y otra oscila sobre el valor cero y es muy pequeña en comparación con los valores de la variable analizada.

Por lo tanto, se puede afirmar que los resultados no se ven alterados por la malla y por tanto la malla es adecuada.

### 4.1.3. Ajustar parámetros en Fluent

Cuando se pulsa el botón *Set up* en Workbench, (ver imagen 70), se abre el programa Fluent. En él han de ajustarse los parámetros que definirán la simulación. Se detallan a continuación pestaña por pestaña. Como se ha usado una versión del programa en anglosajona, se van a utilizar los términos en inglés.

#### 4.1.3.1. General

Se puede decidir qué tipo de *solver* se va a utilizar en la simulación, es decir, las ecuaciones que van a aplicarse, pudiéndose elegir entre *'Pressure-Based Solver'* o '*Density-Based Solver'*.

Ambos solvers se diferencian en la forma en que resuelven la ecuación de continuidad, cantidad de movimiento y energía. El método *Pressure-Based* se utiliza en fluidos incompresibles o medio compresibles. El método *Density-Based* trabaja mejor con fluidos altamente compresibles. En el caso de estudio el fluido es el agua, por lo tanto se utilizará el método *Pressure-Based*.

La simulación será transitoria, *Transient,* y en cuanto a la definición espacial se considerará *Planar.*

#### 4.1.3.2. Models

No se va a aplicar la ecuación de la energía pues no se busca transferencia de calor. Y sólo se definirá la viscosidad como laminar. El programa sólo aplicará las ecuaciones de continuidad y cantidad de movimiento a cada nodo.

#### 4.1.3.3. Materials

El fluido de trabajo es el agua a la que se le asignarán los siguientes valores.

**Tabla 7. Parámetros del fluido.**

| Rs | Densidad [kg/m$^3$] | Viscosidad [kg/(m·s)] |
|---|---|---|
| 70 | 1000 | 0.00035904 |
| 90 | 1000 | 0.00027925 |

#### 4.1.3.4. Boundary Conditions

En este apartado aparecerán los nombres de las partes que se dieron en la malla.

Se debe indicar que se trata de una geometría simétrica. De este modo, cuando se indique que existe un eje de simetría en el dibujo, el propio Fluent simulará el caso como si estuviera duplicado pero sin gastar recursos. Para ello, en *axis-symetry* se elegirá como condición de contorno de tipo simetría. Al llamarse *axis*, Fluent ya entiende que se trata de un eje se simetría y automáticamente le asignará ese tipo de condición de contorno. Del mismo modo, a la pared *Wall* le asignará una condición de tipo Wall. En este caso será estacionaria y sin deslizamiento.

En cuanto a la condición de contorno *Velocity-inlet*, como ya se ha explicado se va a utilizar una velocidad senoidal. Por lo tanto, va a depender del tiempo. Por ello, es necesario crear una UDF y acoplarla con Fluent. En el Anexo E se ha incluido el código. Para acoplar la udf a Fluent en la pestaña superior se seleccionará *Define > User-Defined > Functions > Interpreted* y se seleccionará el archivo donde está escrita la UDF. Después en el apartado *Boundary Conditions > Velocity-Inlet* en la pestaña desplegable se escogerá la opción UDF y la dirección en la que se desea aplicar. En este caso la velocidad oscilatoria se aplica en el eje y.

La ecuación para la velocidad escogida es:

F_PROFILE(f, thread, position) = 0.004*3.1416*sin(2*3.1416*t);

Que proviene de la ecuación [2] del capítulo 2. Donde el valor de la amplitud A es de 0.02 m y la frecuencia utilizada en Fluent siempre es 1.

#### 4.1.3.5. Solution Methods

El momento se resolverá con la opción *Second Order Upwind* por ser la opción que ofrece soluciones más precisas.

#### 4.1.3.6. Monitors

En este apartado sólo se modificarán los criterios de convergencia para que sea del valor de $1 \cdot 10^{-4}$. En vez de $1 \cdot 10^{-3}$ que sería la opción por defecto. Esto significa que se asumirá que la solución ha convergido cuando el error llegue a 0.0001.

#### 4.1.3.7. Run Calculation

Se escogen 24000 *time-steps* es decir, 24000 iteraciones. Y un tamaño de *time-step* de 0.005. Si el periodo es un segundo, quiere decir que un ciclo se ha dividido en 200 divisiones ya que, 1/200 da el tamaño de *time-step*=0.005. Esto quiere decir, que Fluent, resolverá las ecuaciones de Navier-Stokes cada 0.005 segundos.

Se han escogido 200 divisiones porque este valor tiene que ser mucho más pequeño que el tiempo característico, que en este caso es el periodo. Esto es así porque cuantas más divisiones se hagan más datos se obtendrán y por tanto los resultados serán más precisos.

### 4.1.4. Visualización de las imágenes

En Fluent en los dos casos analizados se han realizado 24000 iteraciones. Se le ha pedido a Fluent que exporte ficheros de datos cada 50 time-steps, por lo tanto, se obtienen al final se la simulación 480 ficheros de datos.

Para procesar estos ficheros de datos se ha utilizado un programa llamado Tecplot. Es un software que permite la visualización de programas CFD. Tiene la ventaja de que es compatible con Fluent, por tanto los archivos exportados por Fluent pueden ser leídos directamente mediante Tecplot. Este programa permite analizar estos datos y dibujarlos mediante una interfaz gráfica como la que se muestra a continuación. Ver imágen 75.

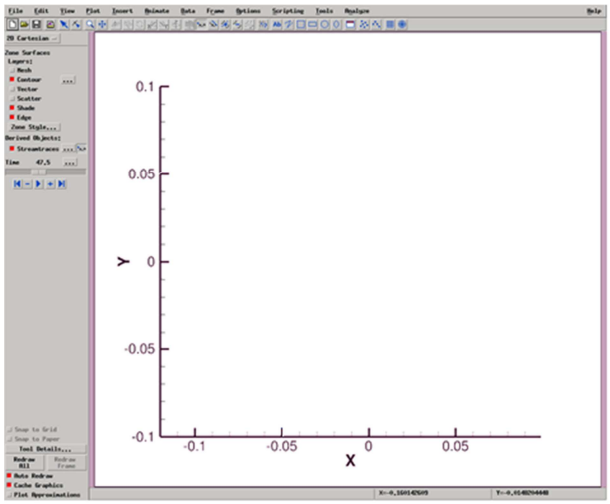



**Imagen 79. Interfaz gráfica de Tecplot.**

Mediante Tecplot se han analizado los 480 archivos de datos y se le ha pedido que muestre las líneas de corriente así como las sendas de las partículas fluidas.

## 4.2. Alternativa a Fluent

Se ha querido plantear una alternativa al programa Fluent. A modo de ejemplo, sólo se plantearán los resultados obtenidos para **Rs=70**, esto es debido a que los resultados apenas varían para **Rs=90**.

Gerris es un software gratuito capaz de resolver las ecuaciones diferenciales que describen el comportamiento de un fluido. A diferencia de otros programas, sólo es posible trabajar con Gerris programando en un lenguaje específico para él. A esto se le llama 'programación basada en consola'. Por otro lado, permite la resolución de problemas complejos con apenas unas líneas de código. Además, es rápido para simular el mismo caso variando sólo algunos parámetros, pues simplemente se cambia en el código el valor sin necesidad de repetir toda la programación. Como contrapunto, ha de añadirse que no es adecuado para modelar flujos turbulentos. Además, para obtener los mismos resultados que con Fluent, es necesario crear una malla con muchos más nodos y por tanto, necesita más tiempo de procesado.

Gerris utiliza un fichero con código como entrada para producir otros ficheros de salida. Si estos ficheros de salida se exportan como ficheros de Gerris (.gfs), se podrán leer con este programa y mediante una interfaz gráfica se podrán ver resultados como líneas de corriente, valores de diferentes variables, etc.

Ha de tenerse en cuenta, que todos los resultados obtenidos mediante esta simulación de Gerris ofrecen una información parcial pues varían con el tiempo. Esta simulación no pretende ofrecer resultados absolutos sino mostrarse como una alternativa a Fluent.

En el anexo F, se incluye el código utilizado por Gerris. A continuación se explican las funciones usadas.

Al comienzo del código se debe especificar de qué tipo de simulación se trata, es decir, el tipo de solver que se quiere usar. *GfsSimulation* es un solver que utiliza las ecuaciones de Euler para flujos incompresibles. Para resolver las ecuaciones de Navier-Stokes existen dos métodos. O bien, al comienzo se utiliza el solver *GfsSourceViscosity* o bien, se utiliza el solver *GfsSimulation* y después, en el código, se añade *SourceViscosity* y a continuación, el nombre de la variable que se utilizará para designar a la viscosidad. Se ha utilizado esta última opción.

A continuación, se definen todas las variables que se va a utilizar, escribiendo *#define* el nombre de la variable y su valor. De esto modo, es fácil simular varios casos donde sólo cambia el valor de las variables, pues el resto del código será igual y simplemente se modificarán estos valores.

Para definir cuándo empieza y cuándo termina la simulación se utiliza la función *GfsTime*.

En cuanto a la malla, se usa *GfsRefine* para definir el número de refinados que tiene que tener la malla inicialmente. El número de refinados, es lo que en el programa se ha llamado '*LEVEL'*. Un ejemplo se muestra en la siguiente imagen.

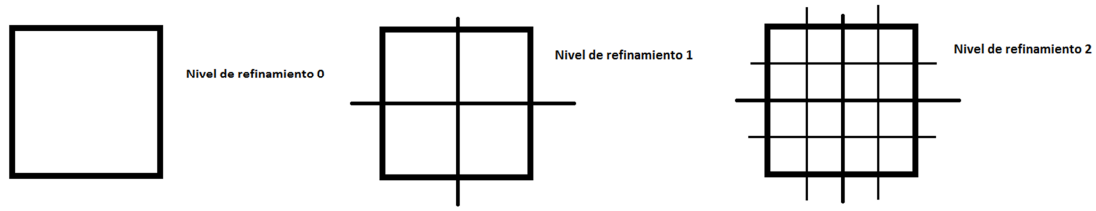



**Imagen 80. Nivel de refinados.**

Para el nivel 0 hay una celda. Para el nivel 1 hay 4 celdas y para el nivel 2 se crean 16. El número de celdas es equivalente a elevar 2 al nivel de refinamiento por dos.

Nº de celdas = $2^{nivel*2}$

Además, se ha utilizado la función *GfsRefineSolid* que refina aquellas celdas que estén en contacto con un sólido. En este caso, se refinarán las celdas que estén en contacto con la superficie del cilindro. Las superficies sólidas se definen mediante *GfsSolid*.

Para conocer el movimiento del fluido, lo que se hace es crear un 'trazador'. Sería parecido a inyectar tinta, de tal modo que ésta se comportase del mismo modo que lo hace el fluido de trabajo. Si no se usara un trazador, en el video que después se exportará se vería todo el fluido igual y no podría distinguirse su movimiento. De esta manera, lo que se va a ver es una parte del fluido de otro color. Para ello, se define la variable trazador, T escribiendo *GfsVariableTracer T*. Y después, la función *GfsInitFraction T*, inicia el movimiento de la variable T alrededor de la superficie que se defina entre corchetes, que para este proyecto, será la pared del sólido.

A continuación, se usa la función *GfsInit* para inicializar variables. En este caso se indica la velocidad tanto en el eje x como en el eje y son cero al comienzo de la simulación.

Después, en el código se ve la función *GfsAdaptFunction* repetida varias veces. Sirve para refinar la malla. Y se puede repetir tantas veces como partes de malla se quieran refinar. Se refinarán las zonas por donde pase el trazador. Además de estos refinados, Gerris ofrece una función capaz de localizar donde se producen gradientes de la variable T (tinta) y coloca más celdas en esas zonas. Esto lo realiza mediante la función *GfsAdaptGradient*. Con la función *GfsAdaptVorticity* realiza la misma operación cuando se producen gradientes en la variable vorticidad.

La malla variará a medida que la simulación evolucione con el tiempo, pues el refinado de la malla varía con la vorticidad y el trazador T. En la imagen 81 se muestra la malla hacia la mitad del tiempo de simulación.

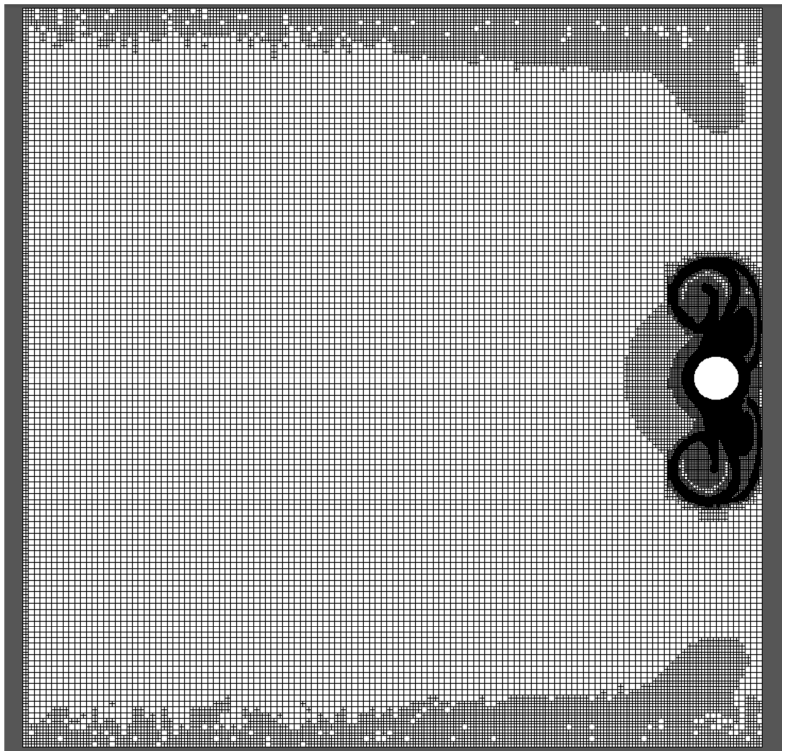


**Imagen 81. Malla Gerris t=160.**

Se ha utilizado la función *GfsEventBalance*, para que la simulación sea más rápida. No afecta al resultado de la simulación, pero divide la carga de trabajo enviándola a distintos procesadores pero reducir el tiempo de la simulación.

Una vez definidas todas las variables, se le ha pedido al programa que realice promedios de variables en el tiempo mediante la función *GfsEventSum*. Es decir, que calcula el valor de esa variable al cabo de un número de iteraciones determinado. Después las almacenará en otras variables. En esta simulación, Gerris se han realizado promedios de la velocidad en el eje x, la velocidad en el eje y la vorticidad. Esta operación la ha realizado cada 10, 20, 30,40, 60, 80, 120, 160, 240 y 320 iteraciones cada vez.

En cuanto a los resultados, se ha pedido a Gerris que exporte algunos parámetros. Mediante *GfsOutputTime* exporta el tiempo de la simulación, cada iteración, el tiempo de la CPU y el tiempo real. *GfsOutputProjectionStats* se utiliza para exportar las estadísticas de convergencia del solver usado. *GfsOutputTiming* resume el tiempo que se ha consumido en la simulación. La función *GfsOutputPPM* exporta una imagen bajo una escala de colores, todas esas imágenes se unirán para crear un video donde se verá el recorrido del trazador. *GfsOutputLocation* escribe todas las variables de un punto seleccionado. *GfsOutputSimulation* exportará una descripción promediada de la simulación, en archivos de Gerris (.gfs). Mediante estos archivos se puede conocer información promediada de por ejemplo las sendas del fluido, las líneas de corriente o las iso-líneas de velocidad. Estos resultados se presentarán más adelante.

El código debe cerrarse con el comando *GfsBox* donde se definen las condiciones de contorno. En esta simulación se recuerda que la velocidad del fluido se rige por un movimiento oscilatorio.

Los archivos exportados por la función *GfsOutputSimulation* se manipularán mediante la interfaz gráfica de Gerris, para obtener los resultados que se muestran en el siguiente apartado. En la siguiente imagen se muestra la interfaz de Gerris.

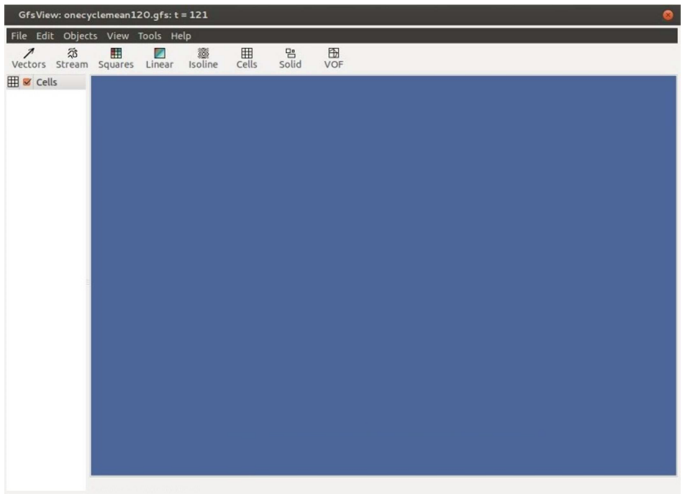



**Imagen 82. Interfaz de Gerris.**

Se trata de un programa muy sencillo de usar donde en la barra de herramientas superior permite dibujar vectores, líneas de corriente, dividir en cuadrados, pintar con colores las variables, dibujar iso-líneas o dibujar figuras sólidas.

Antes de analizar los resultados obtenido con Gerris, se ha observado el video exportado para conocer el movimiento del fluido y poder decir, a priori, si éste, es lógico o directamente se afirma que la simulación es errónea, ya que se conocen las sendas que ha de seguir.

## 4.3. RESULTADOS COMPUTACIONALES

En este capítulo se van a mostrar los resultados obtenidos en las simulaciones por ordenador, tanto los obtenidos por el programa como Fluent por Gerris y se compararán con los resultados obtenidos mediante la experimentación.

### 4.3.1. Fluent

Mediante Tecplot se han analizado los 480 archivos de datos y se le ha pedido que muestre las líneas de corriente así como las sendas de las partículas fluidas.

- Caso **Rs**=70 ;$\epsilon = 0.2$ **;** $\mathbf{g_a}$ = 0.02.

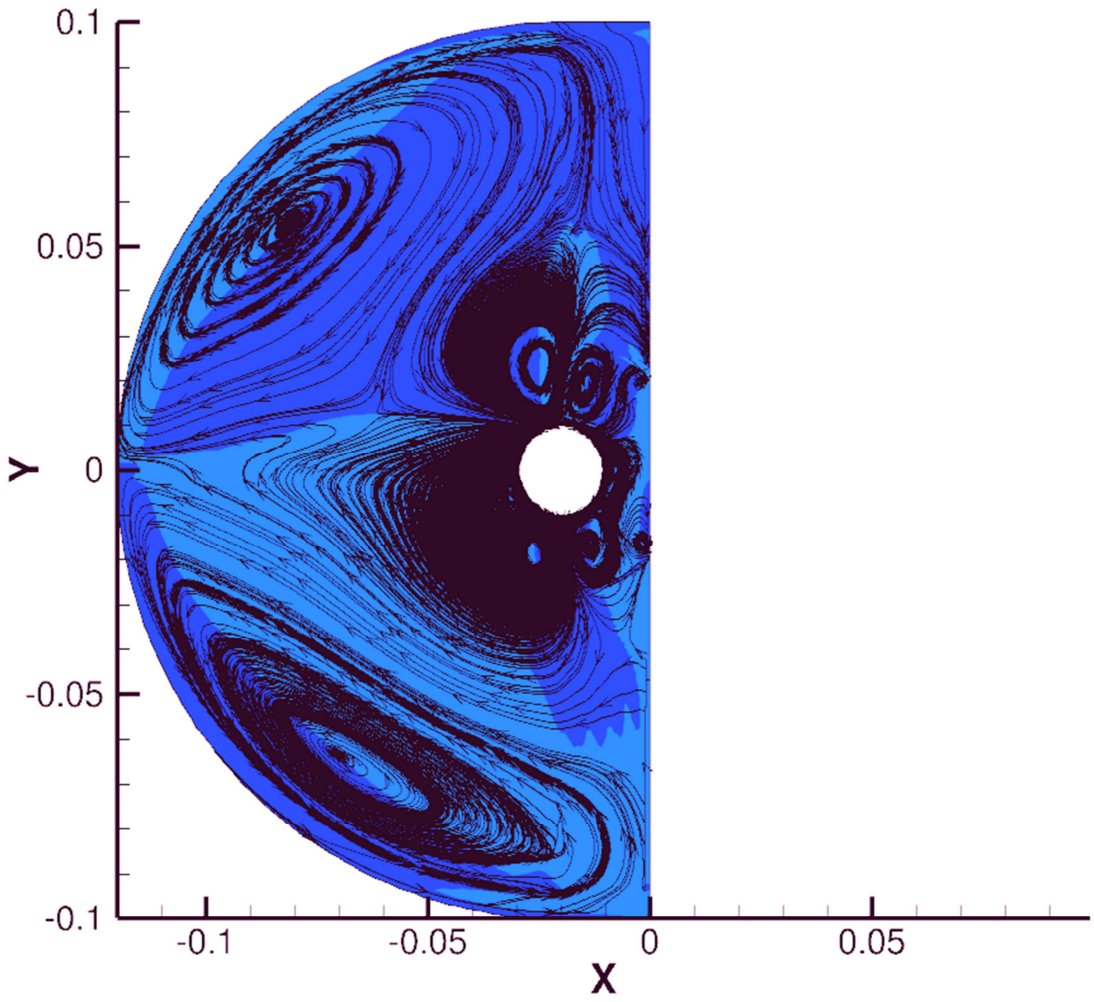



**Imagen 83. Líneas de corriente. Caso Rs=70 ; $\epsilon = 0.2$; ga = 0.02.**

En la imagen 83, se muestran las líneas de corriente. Estas líneas son las mostradas instantáneamente, es decir, en un momento de la simulación concreto, para un tiempo t determinado. Por lo tanto, dan una información relativa pues estas líneas de corriente no

serán igual al principio y al final de la simulación. Las mostradas en la imagen son en un momento intermedio de la simulación. En ellas se puede ver como se forman zonas de recirculación en las zonas cercanas al cilindro

Para obtener una información más exacta, se muestra en la imagen 84, las sendas de las partículas. Es decir, muestra el recorrido que haría una partícula fluida a lo largo de las 24000 iteraciones. Por lo tanto, es independiente del tiempo.

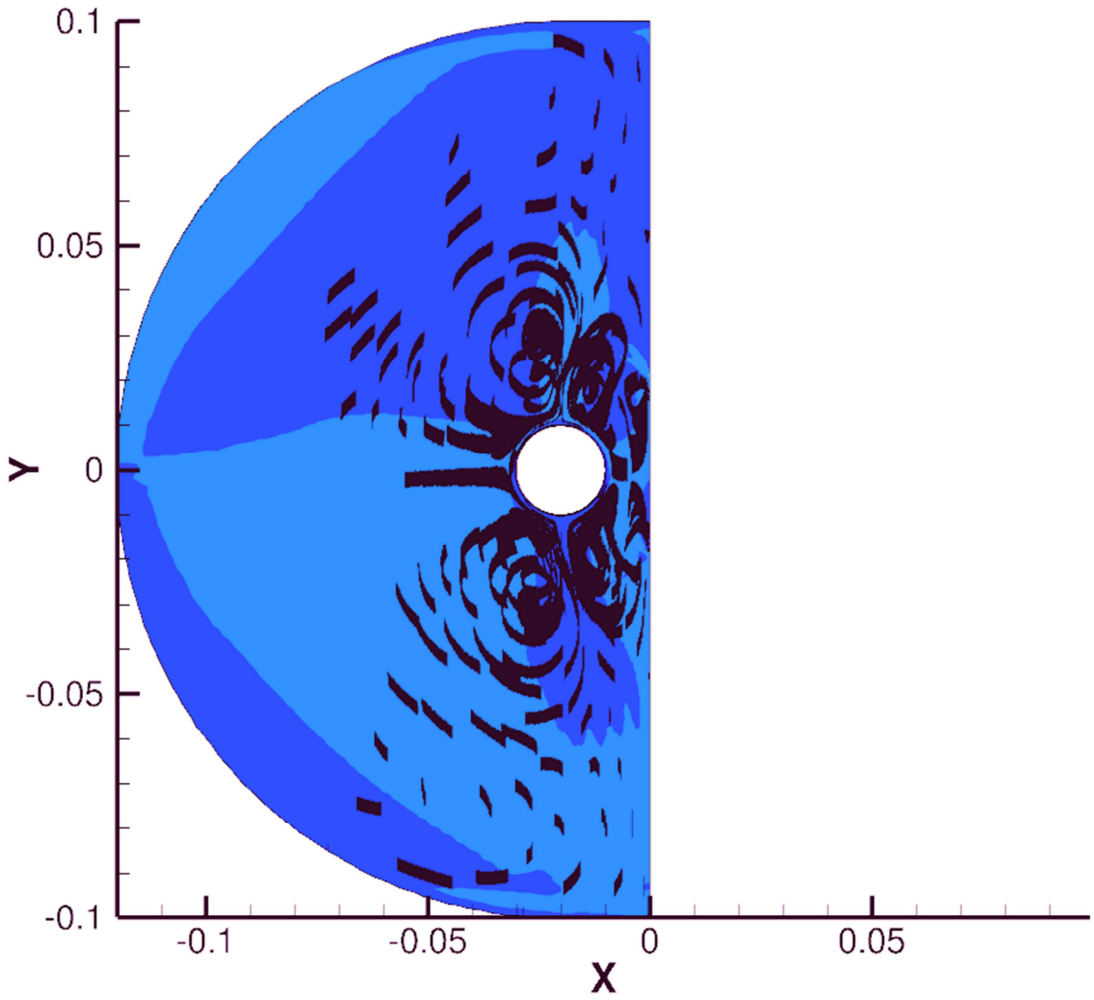



**Imagen 84. Sendas. Caso Rs=70 ; $\epsilon = 0.2$; ga = 0.02.**

De nuevo, se observan las recirculaciones de antes. Nótese, que para que el programa Tecplot realice estos recorridos se le ha de indicar a mano, en qué puntos se desea conocer esta información. Por eso, estas líneas no se ven continuas, por eso también, parece que en las zonas más alejadas las partículas apenas se moverían. La información que se quiere obtener con esta simulación es el detalle que se muestra a mayor tamaño a continuación.

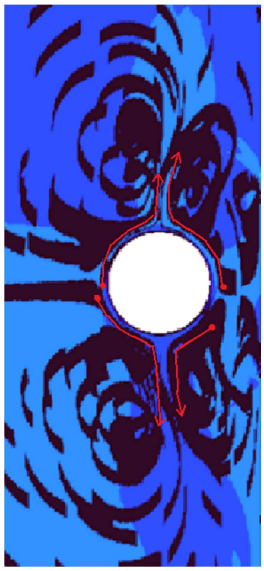


**Imagen 85. Detalle de la recirculación. Caso Rs=70 ; $\epsilon = 0.2$; ga = 0.02.**

Se muestra como una partícula que se deposite cerca de la superficie del cilindro, recorrerá su contorno hasta ser expulsada por el chorro hacía las zonas de recirculación.

- Caso **Rs**=90 ;$\epsilon = 0.2$; $\mathbf{g_a}$ = 0.02.

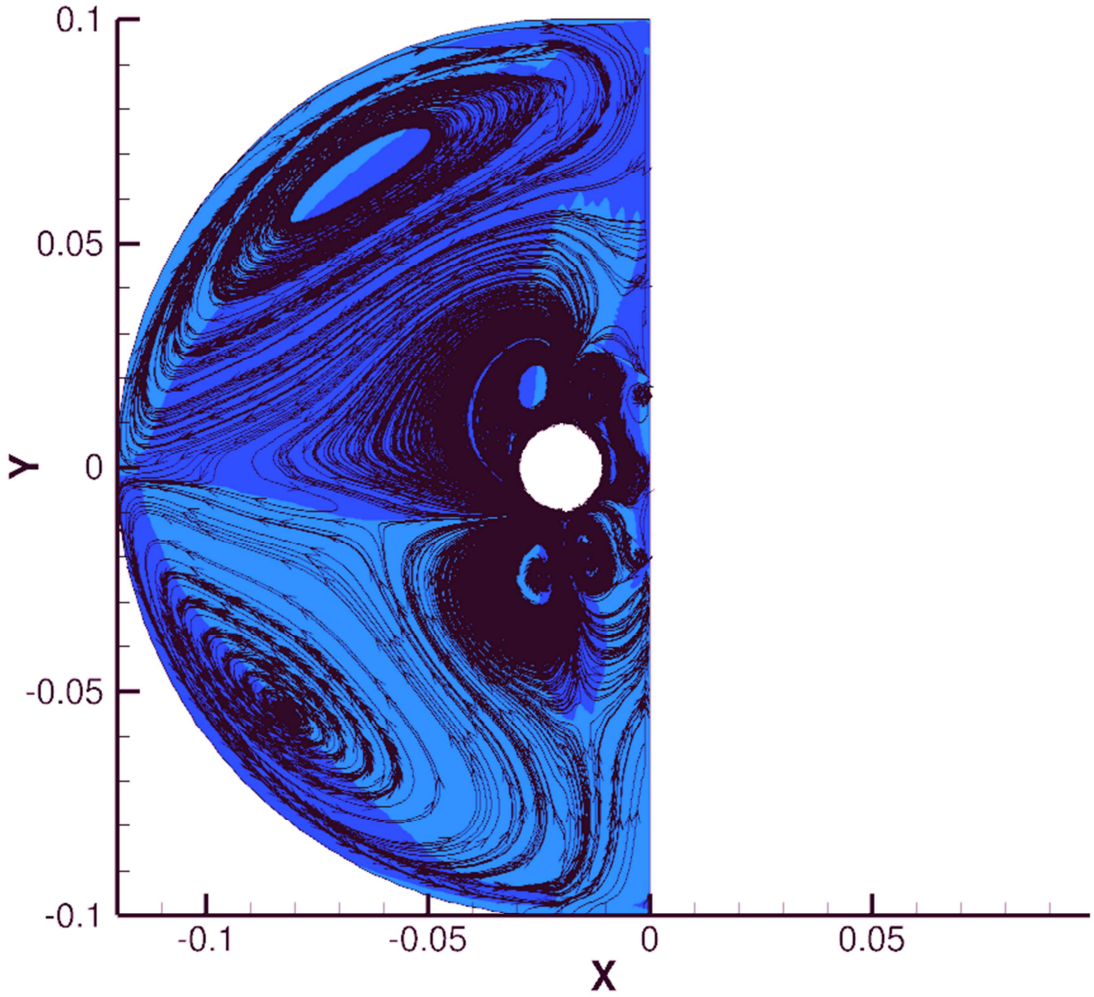



**Imagen 86. Líneas de corriente. Caso Rs=90 ;ϵ=0.2 ; ga = 0.02.**

Apenas existe diferencia entre un caso y el otro, pues los valores de **Rs** son muy parecidos. Ya se ha explicado en el capítulo anterior que aumentar los valores de **Rs** podría convertir el flujo en turbulento y disminuirlos dificultaría la visión de las zonas que se desean estudiar.

Así pues, como antes, se pueden ver las celdas de recirculación.

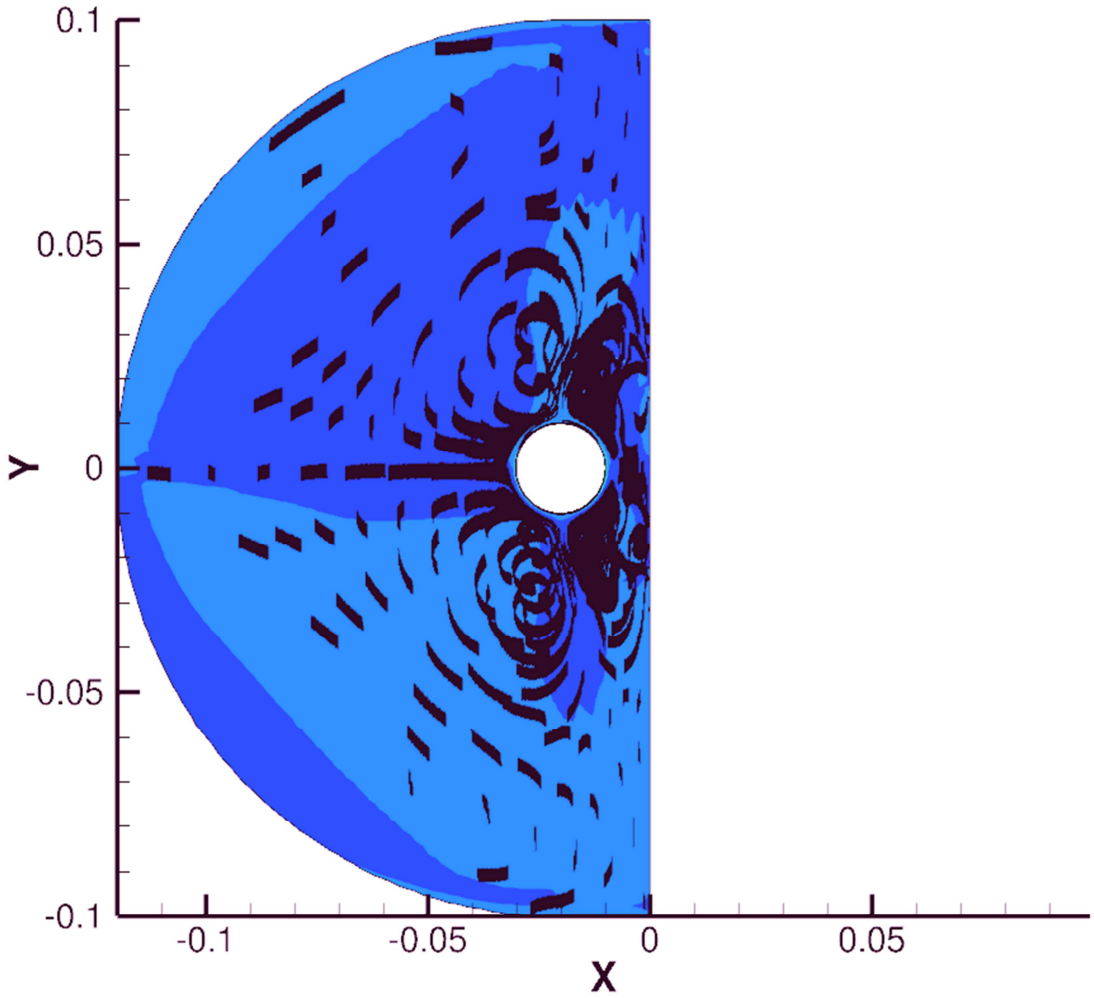



**Imagen 87. Sendas. Caso Rs=90; ϵ=0.2; ga = 0.02.**

En la imagen 87, se ve el recorrido de partículas y de nuevo, se ven que las partículas muy cerca de la pared del cilindro son empujadas al exterior cuando llegan a un determinado ángulo.

### 4.3.2. Comparación entre Fluent y los resultados experimentales

Para dar por buenos los resultados obtenidos por el programa Fluent, se van a comparar con los resultados experimentales. Para realizar esta comparación lo que se ha hecho es comparar ambas imágenes para comprobar que las partículas son eyectadas hacía el exterior de la pared del cilindro y empujadas a las zonas de recirculación.

Para ello, se ha colocado una imagen encima de otra mediante un programa de manipulación de imágenes. El que se ha usado ha sido Photoshop pero valdría cualquiera.

En la imagen 88, se ha colocado la imagen obtenida en experimentos debajo de la obtenida con Tecplot, para el caso 1.

En blanco y negro se ve la imagen real y pintado de rojo se ven las sendas obtenidas por Fluent.

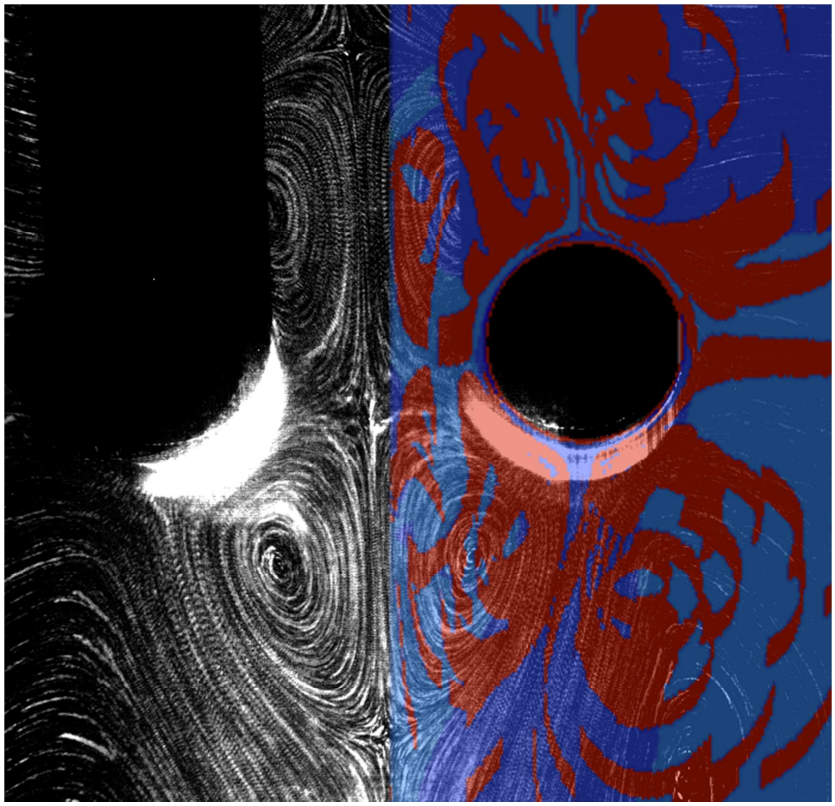


**Imagen 88. Comparación experimental con Fluent para Rs=70.**

Ambos resultados encajan a la perfección pues se ve que los chorros se forman en el mismo punto así como las zonas de recirculación.

Como no se puede ver adecuadamente la similitud entre ambas imágenes se demostrará numéricamente calculando el ángulo del punto de eyección en cada imagen.

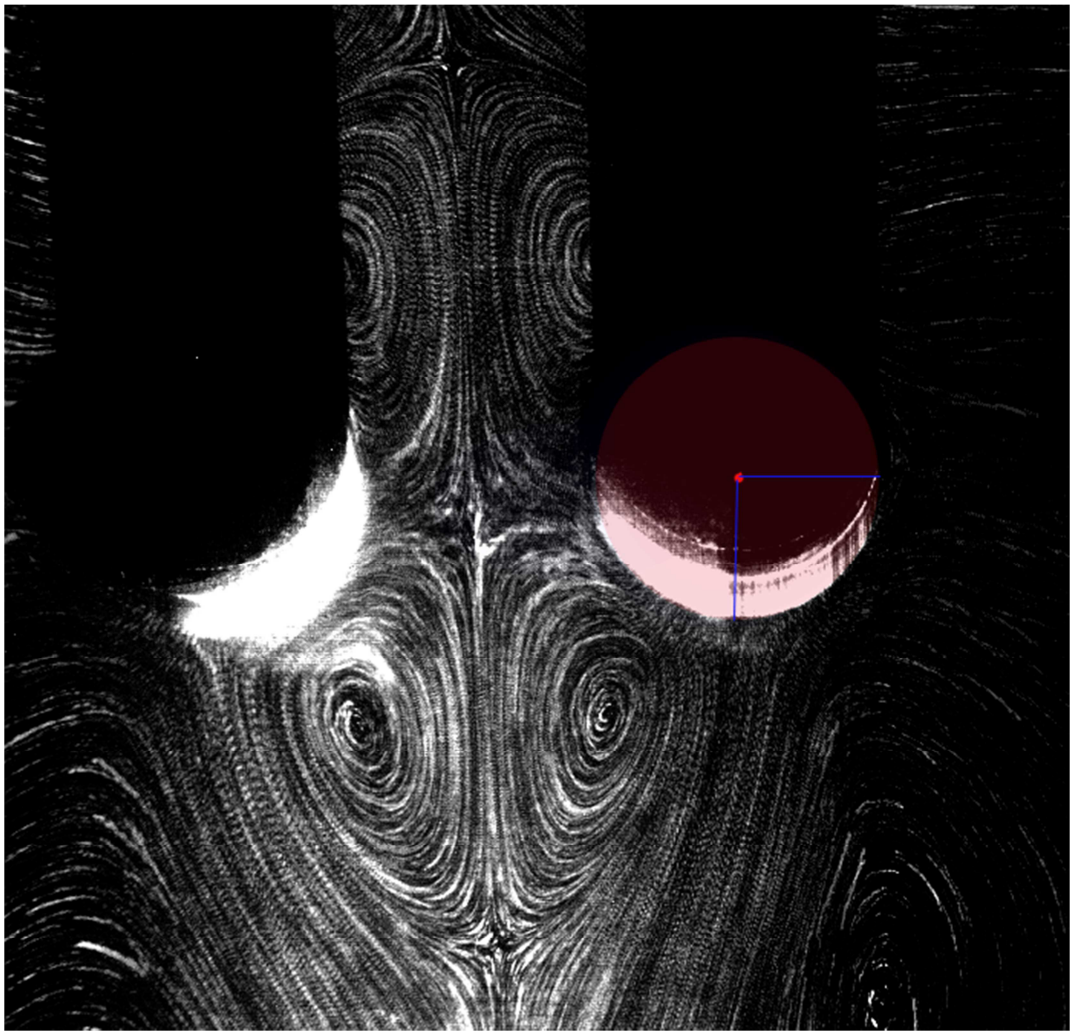


**Imagen 89. Punto de eyección imagen real.**

La línea azul horizontal representa el eje horizontal y la vertical surge de la unión del centro de la circunferencia con el punto de eyección de la partícula. El ángulo que se forma entre estas dos es de 91.5º.

Se repite la operación con la otra imagen.

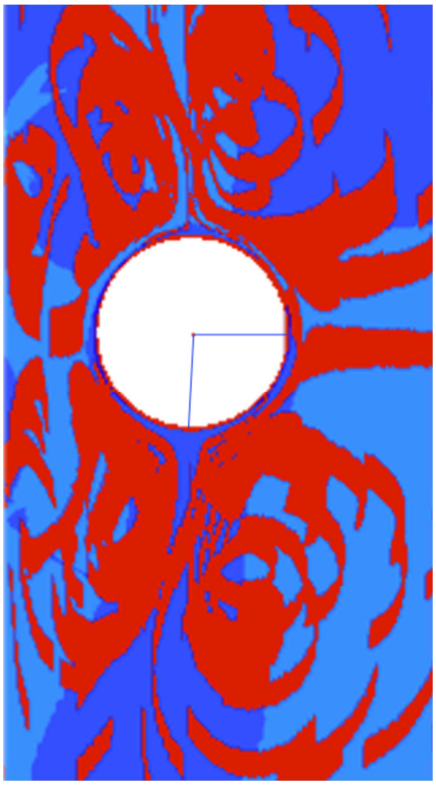


**Imagen 90. Punto de eyección imagen de Fluent. Rs=70.**

En este caso el ángulo entre las dos líneas azules es de 92º.

Se puede asumir que ambos casos coinciden y que por tanto, la simulación con Fluent es válida. Además, se demuestra que el punto de eyección se produce prácticamente a 90º respecto del eje horizontal.

## 4.4. Gerris

En este apartado, se presentan los resultados obtenidos con Gerris para el caso de $\mathbf{Rs} = \mathbf{70}$ ; $\boldsymbol{\epsilon} = \mathbf{0.2}$ ; $\mathbf{g_a} = \mathbf{2a}$.

Usando la interfaz de Gerris se van a pintar las sendas del fluido y después, se compararán con las obtenidas en el experimento del laboratorio mediante Matlab. También, se han obtenido las iso-líneas de velocidad en el eje **x** y en el eje **y**. Además, se ha obtenido el campo de vectores en un tiempo.

Se han realizado promedios de los ciclos cada cierto tiempo. Las imágenes mostradas a continuación son las pertenecientes al promedio de ciclos cuando t=160 segundos hasta t=170 segundos.

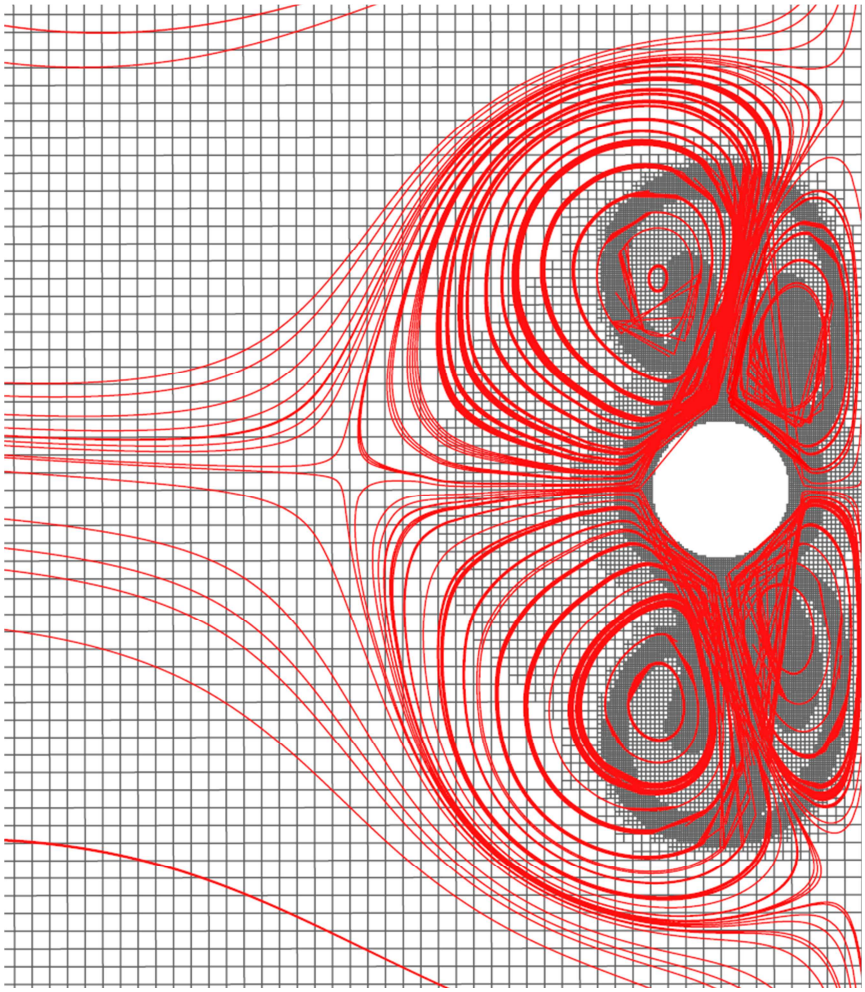


**Imagen 91. Sendas obtenidas por Gerris.**

En la imagen 91, se ven cuatro patrones de recirculación. En las zonas cercanas al eje de simetría estas celdas son más pequeñas que las dos celdas exteriores. Y además, se ve la aparición del chorro y la formación de los dos puntos de eyección correspondientes.

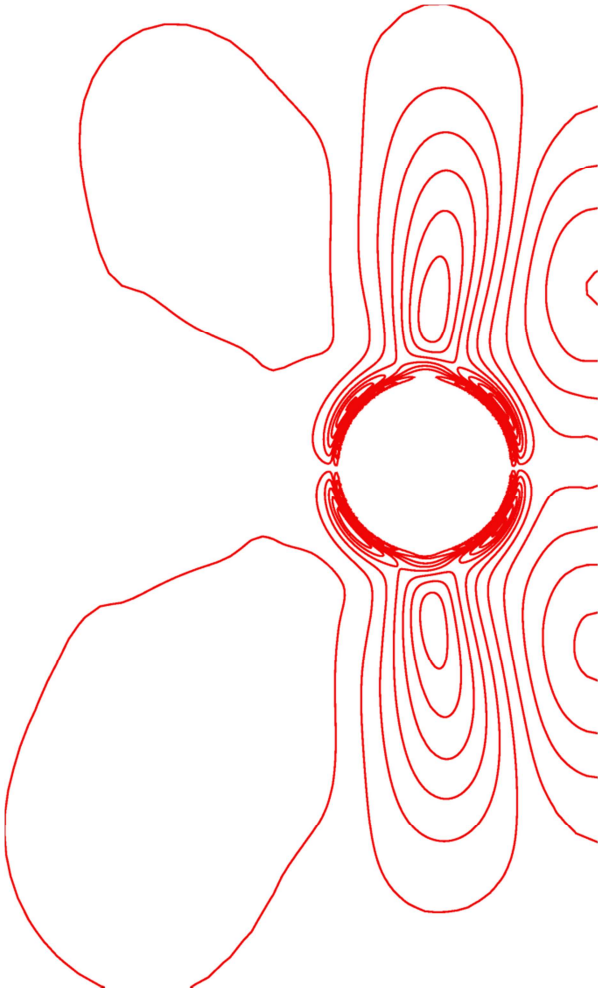


**Imagen 92. Iso-líneas velocidad en el eje y.**

En las iso-líneas de velocidad en el eje y, se observa que, de nuevo, que se producen recirculaciones y que estas no surgen de la superficie sino un poco separadas de los cilindros. En esta imagen los puntos de remanso se ven con claridad. A continuación se muestra un detalle de su posición.

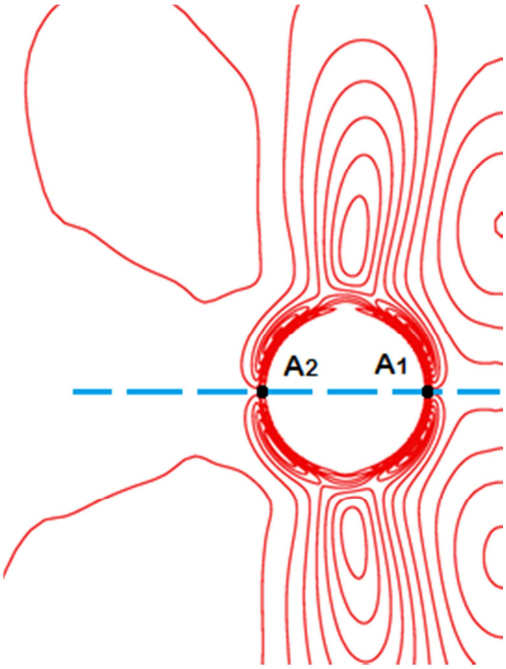



**Imagen 93. Puntos de remanso.**

Además, la línea discontinua representa el eje de coordenadas x, por lo que, los puntos de remanso se encuentran sobre este eje.

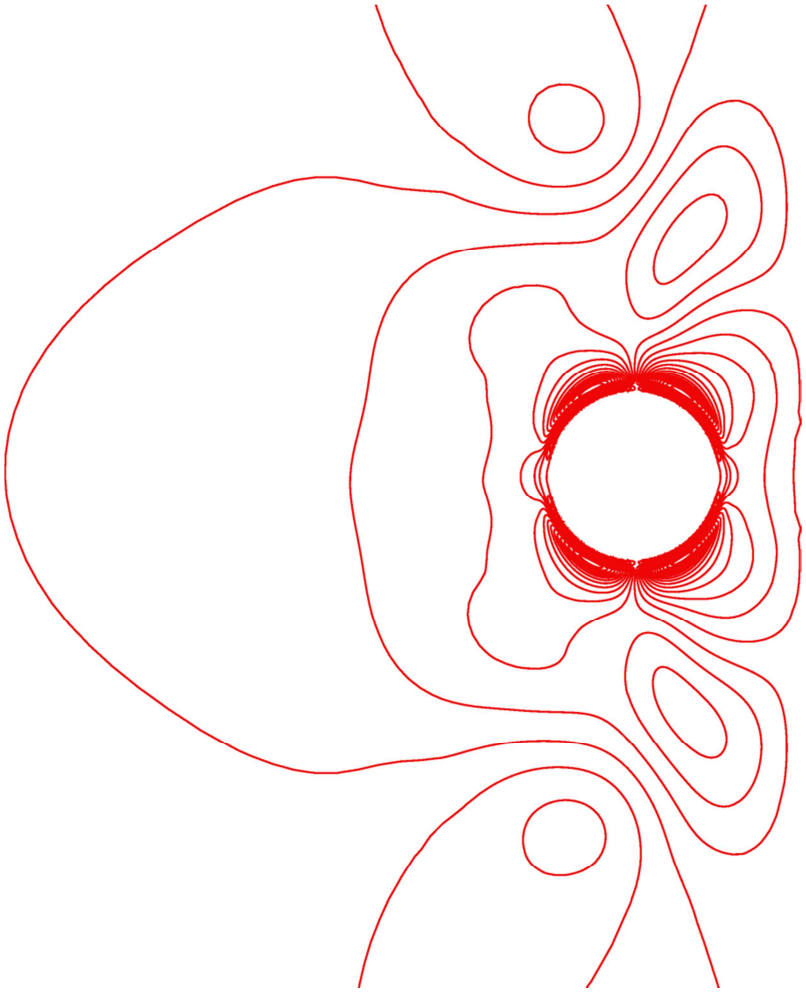


**Imagen 94. Iso-líneas velocidad en el eje x.**

Al contrario que con la imagen anterior, en esta se pueden localizar claramente los puntos de eyección.

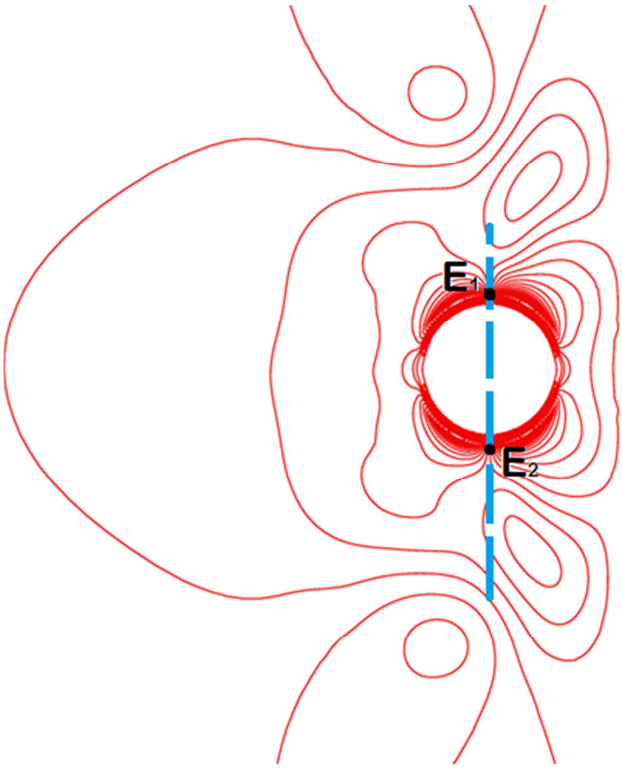



**Imagen 95. Puntos de eyección.**

Los puntos de eyección se encuentran sobre el eje y. Es decir, a 90º del eje horizontal, coincidiendo con lo que se podía observar en las imágenes del laboratorio y las imágenes de Fluent.

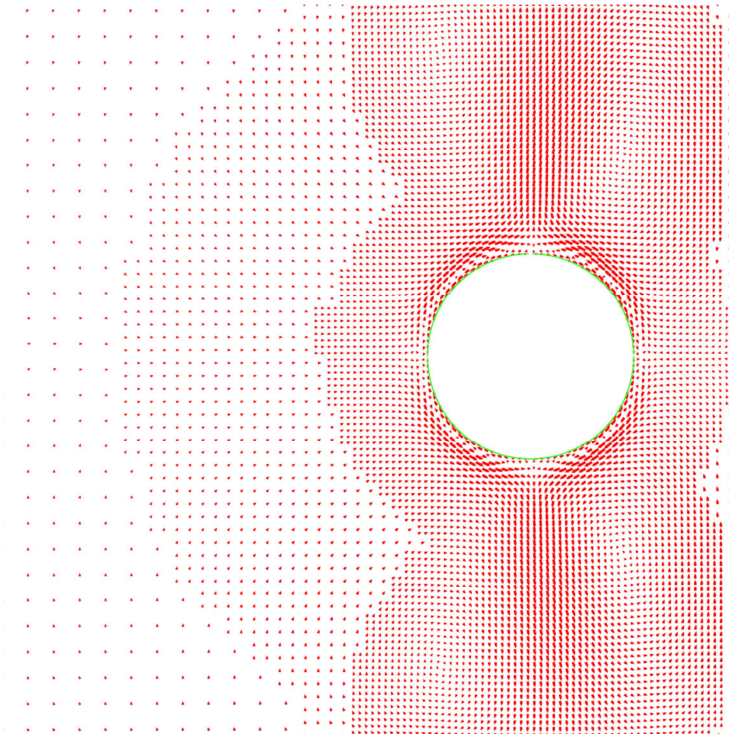


**Imagen 96. Campo de vectores de velocidad.**

Con el campo de velocidades se demuestra que en las zonas de formación del chorro hay más velocidad y también en las zonas más cercanas al cilindro.

### 4.4.1. Comparación entre Gerris y los resultados experimentales

Se ha querido comparar las sendas obtenidas con Gerris con las obtenidas en el laboratorio.

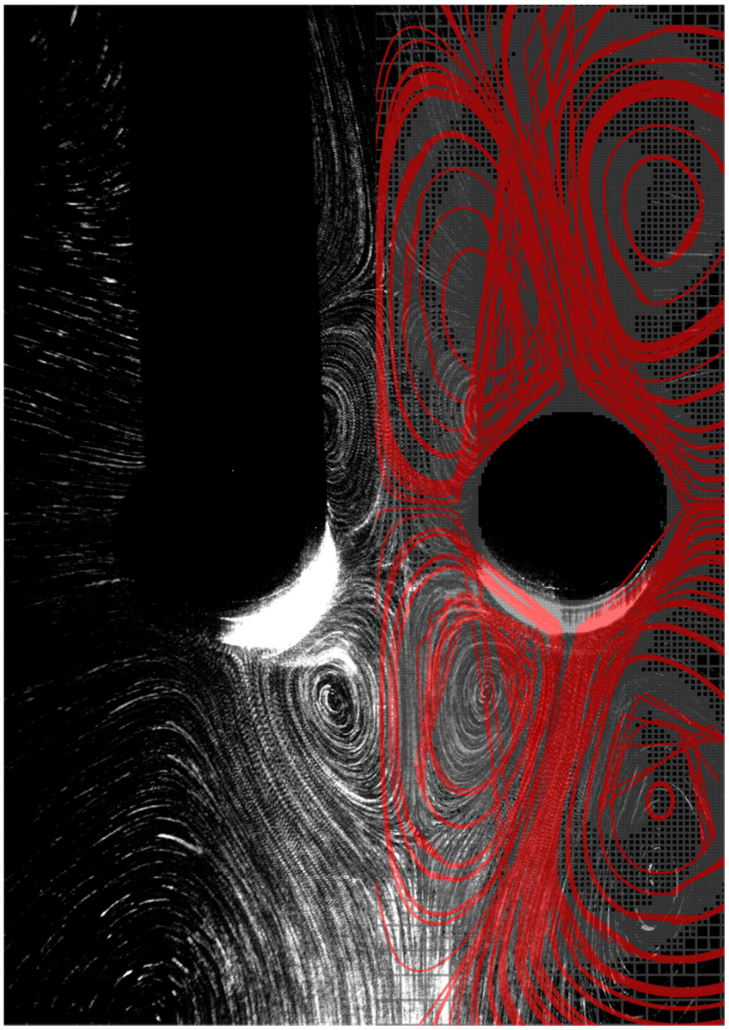


**Imagen 97. Sendas de Gerris frente a las obtenidas experimentalmente.**

Las zonas de recirculaciones en ambos casos, coinciden. Así como la localización de los puntos de eyección.

Además, se ha querido comparar las iso-líneas obtenidas con Matlab frente a las obtenidas con Gerris.

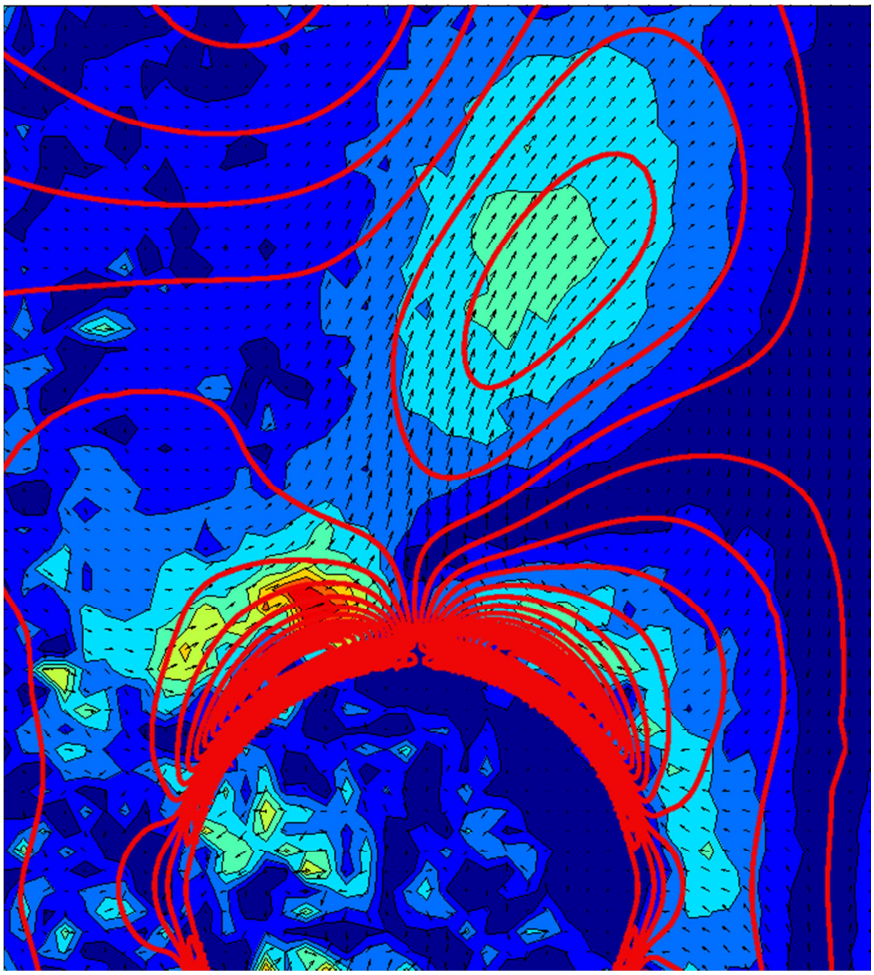


**Imagen 98. Iso-líneas en el eje x. Gerris frente a Matlab.**

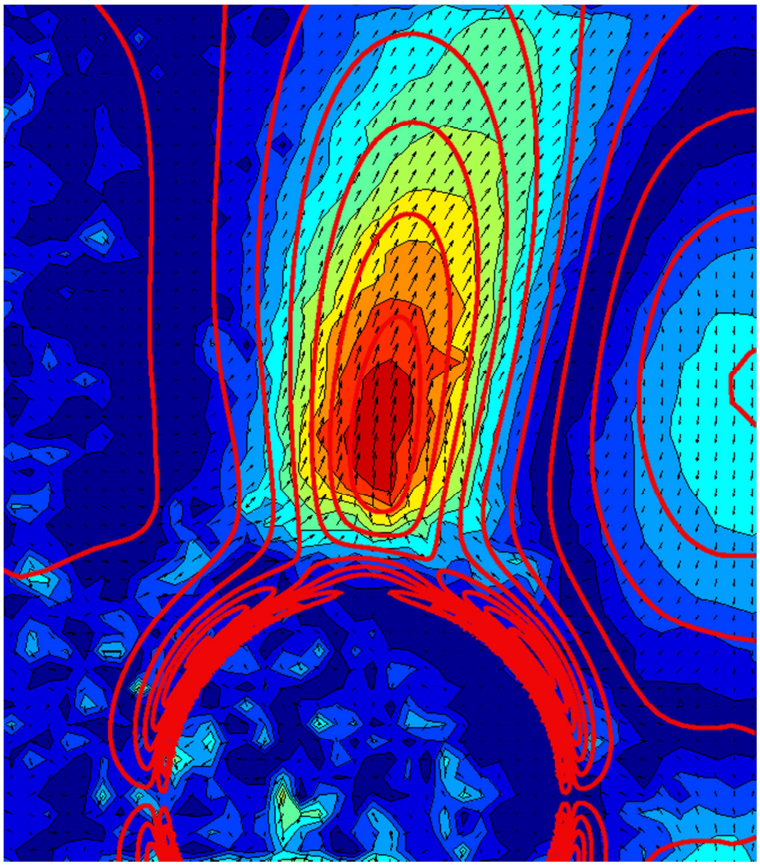


**Imagen 99. Iso-líneas en el eje y. Gerris frente a Matlab.**

En ambos casos la similitud es muy grande. Donde los puntos de remanso y eyección coinciden, así como el sentido del chorro y las celdas de recirculación.

Se puede afirmar, que la simulación realizada mediante Gerris es correcta.

# 5. CONCLUSIONES

# 5. CONCLUSIONES

En este proyecto se ha analizado el efecto *steady streaming* inducido por el movimiento oscilatorio de dos cilindros iguales entre sí para el caso límite en el que se cumple que $\mathbf{Rs} \gg \mathbf{1}$ y $\boldsymbol{\epsilon} \ll \mathbf{1}$. El objetivo de este trabajo ha sido validar las teorías planteadas por **Coenen y Riley** (2009). Para ello, se han realizado experimentos en un laboratorio así como simulaciones por ordenador con dos programas de CFD (*Computational Fluid Dynamics*) distintos, llamados Fluent y Gerris. Para la obtención de conclusiones se han comparados todas los resultados.

Por tanto, para el fenómeno *steady streaming* inducido por el movimiento oscilatorio de dos cilindros cuando se cumple $\boldsymbol{\epsilon} \ll \mathbf{1}$ y $\mathbf{Rs} \gg \mathbf{1}$, se concluye que:

1. La aparición de dos puntos de remanso donde la velocidad es cero y dos puntos de eyección, que surgen de la colisión de dos flujos con direcciones opuestas. En cada punto de eyección se formará un chorro cuya dirección será la opuesta a la superficie del cilindro.

2. Los puntos de remanso se forman en el eje horizontal y los puntos de eyección se forman sobre el eje vertical.

3. El flujo que recorre la superficie del cilindro por la zona más cercana al eje de simetría posee mayor cantidad de movimiento que el flujo que recorre el cilindro por el lado opuesto. A pesar de ello, el chorro irá en dirección al eje de simetría, siendo ambos chorros simétricamente iguales. Por tanto, la dirección del chorro no viene impuesta por el flujo que posee mayor cantidad de movimiento, al contrario que la última hipótesis formulada por **Coenen y Riley**. La dirección de los chorros surge de la interacción entre ambos.

4. Dicha interacción de chorros provoca patrones de recirculación entre ambos cilindros.

5. Debido a la simetría tanto en el eje x como en el eje y, se formarán cuatro celdas de recirculación iguales.

6. Se ha demostrado que, la velocidad en el interior de estas celdas de recirculación es muy pequeña y que las partículas que quedan en el interior de estos vórtices quedan atrapadas.

Se afirma, para terminar, que este tipo de geometría y bajo valores de $\boldsymbol{\epsilon}$ pequeños y números de **Rs** altos, el fenómeno *steady streaming* mecánico es perfecto para utilizarse como **pinzas hidrodinámicas**, tal y como definieron en sus estudios **Lieu et al** (2012).

Estas pinzas podrían utilizarse para usos médicos, tal y como muestran algunos estudios (Lin, Lai, Liu, Chen, & Wo, 2008), que pretenden manipular biopartículas como glóbulos blancos, glóbulos rojos o anticuerpos. Se trata de un método no invasivo que permite la manipulación de estas partículas manteniendo las propiedades originales de éstas durante su manipulación así como reducir el riesgo de una posible contaminación.

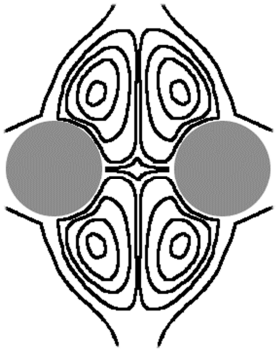


**Imagen 100. Ejemplo de vórtices.**

En la elaboración de este trabajo de fin de grado se han aprendido entre otras, las actividades siguientes:

En el laboratorio:

- Trabajo en equipo con el profesor y los técnicos de laboratorio para el diseño y montaje del experimento
- Ensayo y error como técnica para compensar la falta de experiencia
- Imaginación para resolver las dificultades del día a día
- Uso de una cámara y un láser. Seguridad e higiene en el trabajo
- Compartir recursos y experiencia

En la solución del problema:

- Manejo de artículos y publicaciones
- Explicaciones del profesor sobre algunos modelos, ecuaciones y coeficientes

En la simulación:

- Rigurosidad. La simulación ha de ser controlada.
- Programación de Fluent
- Programación de Gerris
- Programación de Matlab
- Programación en lenguaje C
- Programa PhotoShop
- Uso del *cluster* del departamento

- Uso del sistema operativo Ubuntu

En la redacción:

- Trasladar al documento el trabajo de un año
- Escribir para compañeros que puedan continuar el trabajo sin empezar de cero
- Explicar un trabajo de laboratorio y dos simulaciones en unas pocas páginas

# 6. BIBLIOGRAFÍA

# 6. BIBLIOGRAFÍA

An, C. H., & Zhao, M. (2011). Direct numerical simulation of oscillatory flow around a circular cylinder at low Keulegan-Carpenter number. *Journal of Fluid Mechanics*, 77-103.

Chung, S. C., & Trinh, E. H. (1998). Containerless protein crystal growth in rotating levitated drops. *Journal of Crystal Growth*, 384-397.

Coenen, W., & Riley, N. (2009). Oscillatory flow about a cylinder pair. *Quarterly journal of mechanics and applied mathematics, 62*(1), 53.

Crespo, A. M. (2008). *Mecánica de Fluidos.* Madrid: Thomson.

Davidson, B. J., & Riley, N. (1972). Jets induced by oscillatory motion. *Journal of Fluid Mechanics*, 287-303.

Gopinath, A., & Trinh, E. H. (2000). Compressibility effects on Steady Streaming from a noncompact rigid sphere. *Acoustical Society of America*, 1514-1520.

Honji, H. (Journal of Fluid Mechanics). 1981. *Streaked flow around an oscillating circular cylinder*, 509-520.

Kotas, C. W., Yoda, M., & Rogers, P. H. (2007). Visualization of Steady Streaming near oscillating spheroids. *Experiments in Fluids*, 111-121.

Lieu, V. H., House, T. A., & Schwartz, D. T. (2012). Hydrodynamic tweezers: Impact of design geometry on flow and microparticle trapping. *Analytical Chemistry*, 1963-1968.

Lin, C., Lai, Y., Liu, H., Chen, C., & Wo, A. (2008). Trapping of Bioparticles via Microvortices in a Microfluidic Device for Bioassay Applications. *Analytical chemistry*, 5429-5435.

Nyborg, W. L. (1953). *Acoustical Society of America*, 86.

Otto, F., Riegler, E. K., & Voth, G. A. (2008). Measuremente of the Steady Streaming flow around oscillating spheres usind 3D particle tracking velocimetry. *Physics of Fluids*, 7.

Piercy, J. E., & Lamb, J. (1954). Acoustic Streaming in Liquids. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical, 226*(1164), 43-50.

Riley, N. (2001). Steady Streaming. *Annual Review of Fluid Mechanics*, 43-65.

Riley, N., & Wybrow, M. (1995). The flow induced by the torsional oscillations of an elliptic cylinder. *Journal of Fluid Mechanics*, 408-408.

Wu, J. R., Winkler, A. J., & Oneil, T. P. (1994). Effect of Acoustic Streaming on ultrasonic heating. *Ultrasound in medicine ang biology, 20*, 195-201.

Wybrow, M. F., Yan, B., & Riley, N. (1996). Oscillatory flow over a circular cylinder close to a plane boundary. *Fluid Dynamics Research*, 269-288.

# Anexo A

## Programa buscar frecuencia

```
close all
clear all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Porgrama buscar_freq.m%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Directorio de las imágenes
%imfname =
'../Experimentos/20_03_2013/1b/RawData/1b%06d.T000.D000.P000.H000.LA.T
IF';
%imfname =
'../Experimentos/20_03_2013/1d/RawData/1c%06d.T000.D000.P000.H000.LA.T
IF';
%imfname = '../Experimentos/17_04_2013/2-SS/RawData/2-
SS%06d.T000.D000.P000.H000.LA.TIF';
%imfname = '../Experimentos/17_04_2013/3-SS/RawData/3-
SS%06d.T000.D000.P000.H000.LA.TIF';
imfname =
'../Experimentos/03_05_2013/ss20130503b/img/RawData/img%06d.T000.D000.
P000.H000.LA.TIF';

%Lectura y muestra de imágenes
imn = 1;
im = uint8(imread(sprintf(imfname, imn)));
%imgr = imadjust(im);
%imshow(imgr)
imshow(im)

%Selección de la región de interés
[xroi,yroi] = ginput(2);
xl = round(xroi(1));
yt = round(yroi(1));
hdist = round(xroi(2)-xroi(1));
vdist = round(yroi(2)-yroi(1));
% hold on
% plot(xl,yt,'*')
% plot(xl+hdist,yt+vdist,'s')
% drawnow


for imn = 1:632

   imn
    im = uint8(imread(sprintf(imfname, imn)));
%    im = imadjust(im,[0.5 1.0],[]);
%    imshow(im)
%    hold on
%    plot(xl,yt,'o')
%    plot(xl+hdist,yt+vdist,'s')
%    drawnow
%    pause

%    imgr = imadjust(im);
  %  imshow(imgr)
```

```
   %  gr(imn) = sum(sum(imgr(xl:xl+hdist,yt:yt+vdist)));

    gr(imn) = sum(sum(im(xl:xl+hdist,yt:yt+vdist)));
end

%Resta de la media
gr = gr - mean(gr);

figure(1), plot(gr)
%figure(1), semilogy(gr)

L = length(gr);

Fs = 1;

%Transformada de Fourier
NFFT = 2^nextpow2(L); % Next power of 2 from length of y
Y = fft(gr,NFFT)/L;
f = Fs/2*linspace(0,1,NFFT/2+1);

% Plot single-sided amplitude spectrum.
figure(2), semilogy(f,2*abs(Y(1:NFFT/2+1)))
figure(3), plot(f,2*abs(Y(1:NFFT/2+1)))
```

# Anexo B

## Programa dibujar sendas de las partículas de vidrio

```
clear all
close all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Programa 'sendas_promediado.m %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Directorio de imágenes
imfname =
'../Experimentos/03_05_2013/ss20130503b/img/RawData/img%06d.T000.D000.
P000.H000.LA.TIF';
dirname = '../Experimentos/03_05_2013/ss20130503b/img/b_phases/'
dirphase = '../Experimentos/03_05_2013/ss20130503b/img/b_ultimaimg/'

imini = 1;
imfin = 632;

deltaim = 10.8932;

imf  = imfinfo(sprintf(imfname, imini));
%imav = zeros([imf.Height imf.Width]);
%imdg = logical(zeros(size(imav)));

%glthr = 10;



for k = 1:floor(deltaim)

    vecim = round([k:deltaim:imfin]);  % Devuelve una matriz de imini
hasta imfin con paso de deltaim.
    numim = length(vecim) % devuelve el maximo tamaño de la matriz
    N = 0;
    for imn = 1:numim; %Cada una de las imágenes de cada fase

        im   = uint8(imread(sprintf(imfname, vecim(imn))));

        %figure(1)
        %imshow(im);
        %pause

        %imbg = imopen(im,strel('disk',4));
        %im2  = im - imbg;
        %figure(2)
        %imshow(im2)
        %pause

        % Filtro de ajuste de grises
        im3  = imadjust(im,[0.25 0.75],[]);
%       figure(4)
%       imshow(im4)
%       pause

        %Guardar imágenes
        [dirname,sprintf('%d-%04d',k,imn),'.tif']
     %   imwrite(im,[dirname,sprintf('%d-%04d',k,imn),'.tif']);
        imwrite(im,[dirname,sprintf('%d-%04d',k,imn),'.tif']);
```

```
    %    imshow(im)
    %    pause
%        imbg = imopen(im,strel('disk',4));
%        im2  = im - imbg;
%        im2 = im;
%        im3  = imadjust(im2);
%        imshow(im3)
%        pause


    %   glvl = graythresh(im3)
    %    glvl = 0.95;
    %    imbw = im2bw(im3,glvl);


        if imn == 1
    %        imav = im;
            %imav2 = double(im);
            imav2 = double(im3);
            N = N + 1;
        else

            %imav = imadd(imav,im2uint8(im),'uint8');
    %        imav = imav + im;
    %        imav = imav/2;

            %imav2 = imav2 + double(im);
            imav2 = imav2 + double(im3);


        %    figure(1)
        %    imshow(imav)

        %    imdg = imdg | (im > glthr);

        %    figure(2), imshow(imdg)

        %    pause

        %    imav = imav + double(im);

            N = N + 1;

            %fprintf('Processing image %d...\n', imn);

    %        figure(1)
    %        imshow(imav);
    %        drawnow

%           figure(2)
%           imshow(uint8(imav2/N));
%           drawnow
           % pause

        end

    end
    imav2 = uint8(imav2/N); %Promedio

    imav3  = imadjust(imav2,[0 0.1],[]);    %Ajuste de grises promedio
```

```
    imwrite(imav3,[dirphase,sprintf('%d-%d-
%d',k,imini,imfin),'.tif']);


 figure(3)
 imshow(imav3);
 drawnow


end
```

# Anexo C

## Programa método PIV

```
clear all
close all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Programa 'PIV_promediado.m %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Create list of images inside specified directory
%directory=uigetdir; %directory containing the images you want to
analyze
%directory=['K:/Proyecto Elena/Lab/SS/20_03_2013/exp1a/1d/RawData'];
%directory containing the images you want to analyze
%directory=['/media/TOURO/Proyecto
Elena/Lab/SS/20_03_2013/exp1a/1d/RawData']; %directory containing the
images you want to analyze
directory1 = '../Experimentos/03_05_2013/ss20130503c/img/c_phases';
directory2 = '../Experimentos/03_05_2013/ss20130503d/img/d_phases';
imname1 = '../Experimentos/03_05_2013/ss20130503c/img/c_phases/%d-
%04d.tif';
imname2 = '../Experimentos/03_05_2013/ss20130503d/img/d_phases/%d-
%04d.tif';
variables1 = '../Experimentos/03_05_2013/ss20130503c/img/piv_c/';
variables2 = '../Experimentos/03_05_2013/ss20130503d/img/piv_d/';
pivresult = '../Experimentos/03_05_2013/piv_cd/';

%Lectura del directorio
suffix='*.tif'; %*.bmp or *.tif or *.jpg
direc = dir([directory1,filesep,suffix]); filenames={};
[filenamesall{1:length(direc),1}] = deal(direc.name);
filenames = filenamesall(end-3:end);
filenames = sortrows(filenames); %sort all image files
amount = length(filenames);

% Selección de la región de interés
im=uint8(imread(fullfile(directory1, filenames{1})));
figure(1)
imshow(im)
[xroi,yroi] = ginput(2);
xl = round(xroi(1));
yt = round(yroi(1));
hdist = round(xroi(2)-xroi(1));
vdist = round(yroi(2)-yroi(1));

imc = imcrop(im,[xl, yt, hdist, vdist]);
imshow(imc);

pause
close(1)

    calfac = 1;

        numerodefases =8;      % Valor de deltaim obtenido del
programa 'muestra.m' REDONDEADO A LA BAJA

 for m = 1:2

    for k = 1:numerodefases
```

```
    % elige bien el directorio
    % directory = '...'


    %% Standard PIV Settings
    s = cell(10,2); % To make it more readable, let's create a
"settings table"
    %Parameter                      %Setting          %Options
    s{1,1}= 'Int. area 1';          s{1,2}=64;        % window size
of first pass
    s{2,1}= 'Step size 1';          s{2,2}=32;        % step of
first pass
    s{3,1}= 'Subpix. finder';       s{3,2}=1;         % 1 = 3point
Gauss, 2 = 2D Gauss
    s{4,1}= 'Mask';                 s{4,2}=[];        % If needed,
generate via: imagesc(image); [temp,Mask{1,1},Mask{1,2}]=roipoly;
    s{5,1}= 'ROI';                  s{5,2}=[xl,yt,hdist,vdist]; %
Region of interest: [x,y,width,height] in pixels, may be left empty
    %s{5,1}= 'ROI';                  s{5,2}=[]; % Region of interest:
[x,y,width,height] in pixels, may be left empty
    s{6,1}= 'Nr. of passes';        s{6,2}=3;         % 1-4 nr. of
passes
    s{7,1}= 'Int. area 2';          s{7,2}=32;        % second pass
window size
    s{8,1}= 'Int. area 3';          s{8,2}=16;        % third pass
window size
    s{9,1}= 'Int. area 4';          s{9,2}=16;        % fourth pass
window size
    s{10,1}='Window deformation';   s{10,2}='*spline'; % '*spline' is
more accurate, but slower

    %% Standard image preprocessing settings
    p = cell(8,1);
    %Parameter                      %Setting          %Options
    p{1,1}= 'ROI';                  p{1,2}=s{5,2};    % same as in
PIV settings
    p{2,1}= 'CLAHE';                p{2,2}=1;         % 1 = enable
CLAHE (contrast enhancement), 0 = disable
    p{3,1}= 'CLAHE size';           p{3,2}=20;        % CLAHE window
size
    p{4,1}= 'Highpass';             p{4,2}=1;         % 1 = enable
highpass, 0 = disable
    p{5,1}= 'Highpass size';        p{5,2}=15;        % highpass
size
    p{6,1}= 'Clipping';             p{6,2}=0;         % 1 = enable
clipping, 0 = disable
    p{7,1}= 'Clipping thresh.';     p{7,2}=5;         % 0-255
clipping threshold
    p{8,1}= 'Intensity Capping';    p{8,2}=0;         % 1 = enable
intensity capping, 0 = disable

    %% PIV analysis loop
    if mod(amount,2) == 1 %Uneven number of images?
        disp('Image folder should contain an even number of images.')
        %remove last image from list
        amount=amount-1;
        filenames(size(filenames,1))=[];
    end
    x=cell(amount/2,1);
    y=x;
    u=x;
    v=x;
    typevector=x; %typevector will be 1 for regular vectors, 0 for
masked areas
```

```
    counter=0;
    %% PIV analysis loop:
    if m == 1
         for i=1:2:amount
             counter=counter+1;
         %    image1=imread(fullfile(directory, filenames{i})); % read
images
             image1=imread(sprintf(imname1, k, amount)); % read images
         %    image1=uint8(imread(fullfile(directory, filenames{i})));
% read images
         %    image2=imread(fullfile(directory, filenames{i+1}));
             image2=imread(sprintf(imname1, k, amount+1)); % read
images
         %    image2=uint8(imread(fullfile(directory,
filenames{i+1})));
             image1 = PIVlab_preproc
(image1,p{1,2},p{2,2},p{3,2},p{4,2},p{5,2},p{6,2},p{7,2},p{8,2});
%preprocess images
             image2 = PIVlab_preproc
(image2,p{1,2},p{2,2},p{3,2},p{4,2},p{5,2},p{6,2},p{7,2},p{8,2});
             [x{counter} y{counter} u{counter} v{counter}
typevector{counter}] = piv_FFTmulti
(image1,image2,s{1,2},s{2,2},s{3,2},s{4,2},s{5,2},s{6,2},s{7,2},s{8,2}
,s{9,2},s{10,2});
             clc
             disp([int2str(i/amount*100) ' %']);

             % Graphical output (disable to improve speed)
             %%{
             imagesc(double(image1)+double(image2));colormap('gray');
             hold on

quiver(x{counter},y{counter},u{counter},v{counter},'g','AutoScaleFacto
r', 1.5);
             hold off;
             axis image;
             title(filenames{i},'interpreter','none')
             set(gca,'xtick',[],'ytick',[])
             drawnow;
             %%}
          end
    else
         for i=1:2:amount
             counter=counter+1;
         %    image1=imread(fullfile(directory, filenames{i})); % read
images
             image1=imread(sprintf(imname2, k, amount)); % read images
         %    image1=uint8(imread(fullfile(directory, filenames{i})));
% read images
         %    image2=imread(fullfile(directory, filenames{i+1}));
             image2=imread(sprintf(imname2, k, amount+1)); % read
images
         %    image2=uint8(imread(fullfile(directory,
filenames{i+1})));
             image1 = PIVlab_preproc
(image1,p{1,2},p{2,2},p{3,2},p{4,2},p{5,2},p{6,2},p{7,2},p{8,2});
%preprocess images
             image2 = PIVlab_preproc
(image2,p{1,2},p{2,2},p{3,2},p{4,2},p{5,2},p{6,2},p{7,2},p{8,2});
             [x{counter} y{counter} u{counter} v{counter}
typevector{counter}] = piv_FFTmulti
(image1,image2,s{1,2},s{2,2},s{3,2},s{4,2},s{5,2},s{6,2},s{7,2},s{8,2}
,s{9,2},s{10,2});
             clc
             disp([int2str(i/amount*100) ' %']);
```

```
            % Graphical output (disable to improve speed)
            %%{
            imagesc(double(image1)+double(image2));colormap('gray');
            hold on

quiver(x{counter},y{counter},u{counter},v{counter},'g','AutoScaleFacto
r', 1.5);
            hold off;
            axis image;
            title(filenames{i},'interpreter','none')
            set(gca,'xtick',[],'ytick',[])
            drawnow;
            %%}
        end
    end
    %% PIV postprocessing loop
    % Settings
    umin = -10; % minimum allowed u velocity
    umax = 10; % maximum allowed u velocity
    vmin = -10; % minimum allowed v velocity
    vmax = 20; % maximum allowed v velocity
    stdthresh=7; % threshold for standard deviation check
    epsilon=0.10; % epsilon for normalized median test
    thresh=5; % threshold for normalized median test

    u_filt=cell(amount/2,1);
    v_filt=u_filt;
    typevector_filt=u_filt;

    u_timeavg = zeros(size(u{1,1}));
    v_timeavg = zeros(size(v{1,1}));

    for PIVresult=1:size(x,1)
        u_filtered=u{PIVresult,1};
        v_filtered=v{PIVresult,1};
        typevector_filtered=typevector{PIVresult,1};
        %vellimit check
        u_filtered(u_filtered<umin)=NaN;
        u_filtered(u_filtered>umax)=NaN;
        v_filtered(v_filtered<vmin)=NaN;
        v_filtered(v_filtered>vmax)=NaN;
        % stddev check
        meanu=nanmean(nanmean(u_filtered));
        meanv=nanmean(nanmean(v_filtered));

std2u=nanstd(reshape(u_filtered,size(u_filtered,1)*size(u_filtered,2),
1));

std2v=nanstd(reshape(v_filtered,size(v_filtered,1)*size(v_filtered,2),
1));
        minvalu=meanu-stdthresh*std2u;
        maxvalu=meanu+stdthresh*std2u;
        minvalv=meanv-stdthresh*std2v;
        maxvalv=meanv+stdthresh*std2v;
        u_filtered(u_filtered<minvalu)=NaN;
        u_filtered(u_filtered>maxvalu)=NaN;
        v_filtered(v_filtered<minvalv)=NaN;
        v_filtered(v_filtered>maxvalv)=NaN;
        % normalized median check
        %Westerweel & Scarano (2005): Universal Outlier detection for
PIV data
        [J,I]=size(u_filtered);
        medianres=zeros(J,I);
        normfluct=zeros(J,I,2);
        b=1;
        for c=1:2
```

```
            if c==1; velcomp=u_filtered;else;velcomp=v_filtered;end
%#ok<*NOSEM>
            for i=1+b:I-b
                for j=1+b:J-b
                    neigh=velcomp(j-b:j+b,i-b:i+b);
                    neighcol=neigh(:);

neighcol2=[neighcol(1:(2*b+1)*b+b);neighcol((2*b+1)*b+b+2:end)];
                    med=median(neighcol2);
                    fluct=velcomp(j,i)-med;
                    res=neighcol2-med;
                    medianres=median(abs(res));
                    normfluct(j,i,c)=abs(fluct/(medianres+epsilon));
                end
            end
        end
        info1=(sqrt(normfluct(:,:,1).^2+normfluct(:,:,2).^2)>thresh);
        u_filtered(info1==1)=NaN;
        v_filtered(info1==1)=NaN;

        typevector_filtered(isnan(u_filtered))=2;
        typevector_filtered(isnan(v_filtered))=2;
        typevector_filtered(typevector{PIVresult,1}==0)=0; %restores
typevector for mask

        %Interpolate missing data
        u_filtered=inpaint_nans(u_filtered,4);
        v_filtered=inpaint_nans(v_filtered,4);

        u_filt{PIVresult,1}=u_filtered;
        v_filt{PIVresult,1}=v_filtered;
        typevector_filt{PIVresult,1}=typevector_filtered;

        u_filt{PIVresult,1}=calfac*u_filt{PIVresult,1};
        v_filt{PIVresult,1}=calfac*v_filt{PIVresult,1};

        u_timeavg = u_timeavg + u_filt{PIVresult,1};
        v_timeavg = v_timeavg + v_filt{PIVresult,1};

    end

    u_timeavg = u_timeavg/PIVresult;
    v_timeavg = v_timeavg/PIVresult;

    %clearvars -except amount k p s x y u v typevector directory
filenames u_filt v_filt typevector_filt u_timeavg v_timeavg imc xl yt
hdist vdist

    xx = x{1};
    yy = y{1};

    %[MM,NN] = size(u_timeavg);
    %
    %for i = 1:6
    %    for j = 1:4
    %        u_aa(i,j) = mean2(u_timeavg(floor((i-
1)*MM/6)+1:floor(i*MM/6),floor((j-1)*NN/4)+1:floor(j*NN/4)));
    %        v_aa(i,j) = mean2(v_timeavg(floor((i-
1)*MM/6)+1:floor(i*MM/6),floor((j-1)*NN/4)+1:floor(j*NN/4)));
    %        x_aa(i,j) = mean2(       xx(floor((i-
1)*MM/6)+1:floor(i*MM/6),floor((j-1)*NN/4)+1:floor(j*NN/4)));
    %        y_aa(i,j) = mean2(       yy(floor((i-
1)*MM/6)+1:floor(i*MM/6),floor((j-1)*NN/4)+1:floor(j*NN/4)));
    %    end
```

```
    %end

    %figure(99)
        clf
        hold on
        imagesc(xx(1),yy(1),imc)
        quiver(xx,yy,u_timeavg,v_timeavg,2.0,'r')
    %    quiver(x_aa,y_aa,u_aa,v_aa,'r')
        colormap(gray)
        axis equal
        axis tight
        drawnow
    %    saveas(99,['Results/',caseid,'.fig'])

    %    saveas(99,[writeim,sprintf('phase%d',k),'.tif']);

    %imwrite(figure,[writeim,sprintf('%d-%02d',k,amount),'.tif']);
    %save('resultados') % cambiar esto por un nombre que tenga k

    if k == 1
        uAVG = u_timeavg;
        vAVG = v_timeavg;
    else
        uAVG = uAVG + u_timeavg;
        vAVG = vAVG + v_timeavg;
    end


    end

    %Promedio
    uAVG = uAVG/k;
    vAVG = vAVG/k;

    %figure ()
        clf
        hold on
        imagesc(xx(1),yy(1),imc)
        quiver(xx,yy,uAVG,vAVG,2.0,'r')
        colormap(gray)
        axis equal
        axis tight
        drawnow
 if m == 1
     figure(2)
     save([variables1,'resultspiv.mat'],'xx','yy','uAVG','vAVG')
 else
     figure(3)
     save([variables2,'resultspiv.mat'],'xx','yy','uAVG','vAVG')
 end

end
    %Promedio
    uAVG = uAVG/m;
    vAVG = vAVG/m;

    figure(101)
        clf
        hold on
        imagesc(xx(1),yy(1),imc)
        quiver(xx,yy,uAVG,vAVG,2.0,'r')
        colormap(gray)
        axis equal
        axis tight
```

```
        drawnow

save([pivresult,'resultspiv.mat'],'xx','yy','uAVG','vAVG')
```

# Anexo D

## Programa iso-líneas de velocidad

```
clear all
close all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Programa 'contour_velocity.m' %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

directory = ('../Experimentos/03_05_2013/ss20130503b/img/piv_b/');

%Cargar los datos exportados
load('../Experimentos/03_05_2013/ss20130503b/img/piv_b/resultspiv.mat'
, 'xx');
load('../Experimentos/03_05_2013/ss20130503b/img/piv_b/resultspiv.mat'
, 'yy');
load('../Experimentos/03_05_2013/ss20130503b/img/piv_b/resultspiv.mat'
, 'uAVG');
load('../Experimentos/03_05_2013/ss20130503b/img/piv_b/resultspiv.mat'
, 'vAVG');

%Convertir a vector
xx = xx(1,:)
yy = yy(:,1)

numpixel = 187; % Creo y sería aproximado

% Factores de conversión para m/s
uvel = (uAVG * 2 * 0.01 * 30)/numpixel;
vvel = (vAVG * 2 * 0.01 * 30)/numpixel;

xxms = (xx*2*0.01/numpixel - 0.06565)*100;
yyms = (yy*2*0.01/numpixel - 0.065)*100;

uvel = 100*uvel; % en cm/s
vvel = 100*vvel; % en cm/s

v = sqrt(uvel.^2 + vvel.^2);

% Diagramas de iso-líneas

figure (1)
hold on
contourf (xxms,yyms,abs(uvel))
quiver(xxms,yyms,uvel,vvel,'k-')
axis equal
axis tight
xlabel('cm')
ylabel('cm','rotation',0)
hcb = colorbar;
ylabel(hcb,'cm/s','rotation',0);

figure (2)
hold on
contourf (xxms,yyms,abs(vvel))
quiver(xxms,yyms,uvel,vvel,'k-')
axis equal
axis tight
xlabel('cm')
ylabel('cm','rotation',0)
```

```
hcb = colorbar;
ylabel(hcb,'cm/s','rotation',0);

figure (3)
hold on
contourf (xxms,yyms,v)
quiver(xxms,yyms,uvel,vvel,'k-')
axis equal
axis tight
xlabel('cm')
ylabel('cm','rotation',0)
hcb = colorbar;
ylabel(hcb,'cm/s','rotation',0);
```

# Anexo E

## Programa UDF velocidad variable en las condiciones de contorno

```
/************************************************************************
*
Programa 'unsteadybc.c'
UDF perfil de velocidad dependiente del tiempo
************************************************************************
*/

#include "udf.h"

DEFINE_PROFILE(unsteady_velocity, thread, position)
{
face_t f;
real t = CURRENT_TIME;

begin_f_loop(f, thread)
{
F_PROFILE(f, thread, position) = 0.004*3.1416*sin(2*3.1416*t);
}
end_f_loop(f, thread)
}
```

# Anexo F

## Programa Simulación Gerris

```
############################
# Código programa Gerris #
############################

1 0 GfsSimulation GfsBox GfsGEdge {} {

    Global {

    #define pi      3.14159265358979323846

    #define phi     (pi/2.0)
    #define gap     2.0
    #define Rs      70.0
    #define epsilon 0.2

    #define a1      0.03125
    #define cc1     (0.5-0.5*gap*a1-a1)

    #define LEVEL   14

    #define freq    1.0

    #define U0      (2*pi*freq*epsilon*a1)
    #define nu      (epsilon*a1*U0/Rs)

    #define mu      nu

    #define slt     (epsilon*a1/sqrt(Rs))
    #define blt     (a1/sqrt(Rs))

    }

    Time { end = 321.0 dtmax = 0.01 }

    Refine (LEVEL-7)

    Solid ( ellipse(cc1, 0.0, a1, a1) )

    RefineSolid LEVEL

    VariableTracer T

    InitFraction T ({ return ( - difference( ellipse( cc1, 0.0,
a1+blt, a1+blt),
                                        ellipse( cc1, 0.0, a1,
a1    ) ) ); })

    Init {} { U = 0.0  V = 0.0 }

    AdaptFunction { istep = 1 } { cmax = 0.01 minlevel = (LEVEL-7)
maxlevel = (LEVEL-0) } ( (x-cc1)*(x-cc1) + y*y <
(1.031250*a1)*(1.031250*a1) )
    AdaptFunction { istep = 1 } { cmax = 0.01 minlevel = (LEVEL-7)
maxlevel = (LEVEL-1) } ( (x-cc1)*(x-cc1) + y*y <
(1.062500*a1)*(1.062500*a1) )
```

```
    AdaptFunction { istep = 1 } { cmax = 0.01 minlevel = (LEVEL-7)
maxlevel = (LEVEL-2) } ( (x-cc1)*(x-cc1) + y*y <
(1.125000*a1)*(1.125000*a1) )
    AdaptFunction { istep = 1 } { cmax = 0.01 minlevel = (LEVEL-7)
maxlevel = (LEVEL-3) } ( (x-cc1)*(x-cc1) + y*y <
(1.250000*a1)*(1.250000*a1) )
    AdaptFunction { istep = 1 } { cmax = 0.01 minlevel = (LEVEL-7)
maxlevel = (LEVEL-4) } ( (x-cc1)*(x-cc1) + y*y <
(1.500000*a1)*(1.500000*a1) )
    AdaptFunction { istep = 1 } { cmax = 0.01 minlevel = (LEVEL-7)
maxlevel = (LEVEL-5) } ( (x-cc1)*(x-cc1) + y*y <
(2.000000*a1)*(2.000000*a1) )
    AdaptFunction { istep = 1 } { cmax = 0.01 minlevel = (LEVEL-7)
maxlevel = (LEVEL-6) } ( (x-cc1)*(x-cc1) + y*y <
(4.000000*a1)*(4.000000*a1) )

    AdaptGradient  { istep = 1 } { cmax = 0.01 maxlevel = ( (x-
cc1)*(x-cc1) + y*y > 8.0*a1*8.0*a1 ? (LEVEL-6) : (LEVEL-1) ) } T
    AdaptVorticity { istep = 1 } { cmax = 0.01 maxlevel = ( (x-
cc1)*(x-cc1) + y*y > 8.0*a1*8.0*a1 ? (LEVEL-6) : (LEVEL-2) ) }

    SourceViscosity mu

    EventBalance { istep = 100 } 0.1

    EventSum { start =  10.0 end =  10.995 step = 0.005 }         U
Um010
    EventSum { start =  10.0 end =  10.995 step = 0.005 }         V
Vm010
    EventSum { start =  10.0 end =  10.995 step = 0.005 } Vorticity
vort010

    EventSum { start =  20.0 end =  20.995 step = 0.005 }         U
Um020
    EventSum { start =  20.0 end =  20.995 step = 0.005 }         V
Vm020
    EventSum { start =  20.0 end =  20.995 step = 0.005 } Vorticity
vort020

    EventSum { start =  30.0 end =  30.995 step = 0.005 }         U
Um030
    EventSum { start =  30.0 end =  30.995 step = 0.005 }         V
Vm030
    EventSum { start =  30.0 end =  30.995 step = 0.005 } Vorticity
vort030

    EventSum { start =  40.0 end =  40.995 step = 0.005 }         U
Um040
    EventSum { start =  40.0 end =  40.995 step = 0.005 }         V
Vm040
    EventSum { start =  40.0 end =  40.995 step = 0.005 } Vorticity
vort040

    EventSum { start =  60.0 end =  60.995 step = 0.005 }         U
Um060
    EventSum { start =  60.0 end =  60.995 step = 0.005 }         V
Vm060
    EventSum { start =  60.0 end =  60.995 step = 0.005 } Vorticity
vort060

    EventSum { start =  80.0 end =  80.995 step = 0.005 }         U
Um080
    EventSum { start =  80.0 end =  80.995 step = 0.005 }         V
Vm080
```

```
    EventSum { start =  80.0 end =  80.995 step = 0.005 } Vorticity
vort080

    EventSum { start = 120.0 end = 120.995 step = 0.005 }         U
Um120
    EventSum { start = 120.0 end = 120.995 step = 0.005 }         V
Vm120
    EventSum { start = 120.0 end = 120.995 step = 0.005 } Vorticity
vort120

    EventSum { start = 160.0 end = 160.995 step = 0.005 }         U
Um160
    EventSum { start = 160.0 end = 160.995 step = 0.005 }         V
Vm160
    EventSum { start = 160.0 end = 160.995 step = 0.005 } Vorticity
vort160

    EventSum { start = 240.0 end = 240.995 step = 0.005 }         U
Um240
    EventSum { start = 240.0 end = 240.995 step = 0.005 }         V
Vm240
    EventSum { start = 240.0 end = 240.995 step = 0.005 } Vorticity
vort240

    EventSum { start = 320.0 end = 320.995 step = 0.005 }         U
Um320
    EventSum { start = 320.0 end = 320.995 step = 0.005 }         V
Vm320
    EventSum { start = 320.0 end = 320.995 step = 0.005 } Vorticity
vort320

    OutputTime            { istep =   1 } log.out
    OutputProjectionStats { istep =   1 } log.out
    OutputTiming          { istep = 100 } log.out
    OutputBalance         { istep =   1 } log.out

    OutputSimulation { step = 1.00 } ss---%010.5f.gfs { }
#   OutputSimulation { step = 1.00 } ss---%010.5f.vtk { format=VTK }

#   OutputPPM { step = 1.00 }  vort---%010.5f.ppm   { min =  -10.00
max =  10.00 v = Vorticity maxlevel = 12 }
#   OutputPPM { step = 1.00 }    T1---%010.5f.ppm   { min =    0.00
max =   1.00 v = T         maxlevel = 12 }
#   OutputPPM { step = 1.00 }    T2---%010.5f.ppm   { min =    0.00
max =   0.30 v = T         maxlevel = 12 }
#   OutputPPM { step = 1.00 }    T3---%010.5f.ppm   { min =    0.00
max =   0.10 v = T         maxlevel = 12 }

#   OutputPPM { step = 0.05 }              tracer.ppm   { min = 0.00
max = 1.00 v = T maxlevel = 12 condition = ( x > 0.0 && x < 0.5 && y >
-0.250 && y < 0.250 ) }
    OutputPPM { step = 0.05 } { ppm2mpeg > tracer.mpg } { min = 0.00
max = 1.00 v = T maxlevel = 12 condition = ( x > 0.0 && x < 0.5 && y >
-0.250 && y < 0.250 ) }

    OutputLocation { istep = 1 } outputatlowerbnd  0.000 -0.499  0.0
    OutputLocation { istep = 1 } outputatupperbnd  0.000  0.499  0.0
    OutputLocation { istep = 1 } outputatleftbnd  -0.499  0.000  0.0
    OutputLocation { istep = 1 } outputatrightbnd  0.499  0.000  0.0

    OutputSimulation { start =   11.0 } onecyclemean010.gfs {
variables = Um010,Vm010,vort010 }
    OutputSimulation { start =   21.0 } onecyclemean020.gfs {
variables = Um020,Vm020,vort020 }
```

```
    OutputSimulation { start =   31.0 } onecyclemean030.gfs {
variables = Um030,Vm030,vort030 }
    OutputSimulation { start =   41.0 } onecyclemean040.gfs {
variables = Um040,Vm040,vort040 }
    OutputSimulation { start =   61.0 } onecyclemean060.gfs {
variables = Um060,Vm060,vort060 }
    OutputSimulation { start =   81.0 } onecyclemean080.gfs {
variables = Um080,Vm080,vort080 }
    OutputSimulation { start =  121.0 } onecyclemean120.gfs {
variables = Um120,Vm120,vort120 }
    OutputSimulation { start =  161.0 } onecyclemean160.gfs {
variables = Um160,Vm160,vort160 }
    OutputSimulation { start =  241.0 } onecyclemean240.gfs {
variables = Um240,Vm240,vort240 }
    OutputSimulation { start =  321.0 } onecyclemean320.gfs {
variables = Um320,Vm320,vort320 }

}

GfsBox {
  top    = Boundary { BcDirichlet U U0*sin(2*pi*freq*t)*cos(phi)
BcDirichlet V U0*sin(2*pi*freq*t)*sin(phi) }
  bottom = Boundary { BcDirichlet U U0*sin(2*pi*freq*t)*cos(phi)
BcDirichlet V U0*sin(2*pi*freq*t)*sin(phi) }
  left   = Boundary { BcDirichlet U U0*sin(2*pi*freq*t)*cos(phi)
BcDirichlet V U0*sin(2*pi*freq*t)*sin(phi) }
  right  = Boundary
}
```